Implementation of a cooperative communication within a fleet of connected and autonomous vehicles

Guilhem Marcillaud

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse III - Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *01/02/2022* **par :**

Guilhem MARCILLAUD

Implementation of a Cooperative Communication within a Fleet of Connected and Autonomous Vehicles

---

JURY

MARIE-PIERRE GLEIZES	Professeur d'Université	Co-Directrice
VALÉRIE CAMPS	Maître de Conference	Co-Directrice
RENÉ MANDIAU	Professeur d'Université	Rapporteur
MAHDI ZARGAYOUNA	Chargé de Recherche	Rapporteur
OLIVIER SIMONIN	Professeur d'Université	Examinateur
STÉPHANIE COMBETTES	Maître de Conference	Invitée

École doctorale et spécialité :
MITT : Domaine Mathématiques : Mathématiques appliquées
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (IRIT)
Directeur(s)/Encadrant(s) de Thèse :
Valérie Camps, *Stéphanie Combettes* et *Marie-Pierre Gleizes*
Rapporteurs :
René Mandiau et *Mahdi Zargayouna*

Guilhem Marcillaud

IMPLEMENTATION OF A COOPERATIVE COMMUNICATION WITHIN A FLEET OF AUTONOMOUS AND CONNECTED VEHICLES

Thesis Supervisors **Marie Pierre Gleizes**, Full Professor, Université Toulouse III Paul Sabatier
Valérie Camps, Associate Professor, Université Toulouse III Paul Sabatier
Stéphanie Combettes, Associate Professor, Université Toulouse III Paul Sabatier

THIS thesis addresses the problem of communication in the context of a fleet of autonomous and connected vehicles. It is included in the context of intelligent transport systems and the smart city. Numerous applications have been developed in recent years in these areas, each with the objective of improving the quality of life of users. The automation of driving, which has been a central concern of the automotive industry since the 2000s, is expected to reduce the number of accidents, improve the comfort of users and also reduce the ecological footprint of road traffic in general. It also paves the way for the implementation of effective cooperative strategies between vehicles. As cooperation relies on the exchange of relevant information, it is necessary that vehicles are able to know what information to transmit and how.

This thesis focuses on inter-vehicle communication and approaches the problem as a complex system in which many vehicles interact with each other, each has its own local objective. Each vehicle perceives information in its local environment and knows that some of this information may be useful to neighbouring vehicles. As a cooperative entity, it shares with its neighbours the information it assumes to be useful. In this work, an inter-vehicle communication is considered useful if it verifies the following two properties: 1) the information exchanged is understood by the receiving vehicle and 2) it brings new knowledge to it. In the context of a fleet of Connected and Autonomous Vehicles (CAVs), these two properties may not always be guaranteed, especially if the vehicles involved do not share the same referential frame (e.g. different units of measurement for the same information) or if the volume of communication exceeds the vehicle's capacity.

The contribution of this thesis is twofold: it proposes a first module allowing a vehicle to adapt information to its own reference system, and a second module allowing to optimise information exchanges within a fleet of CAVs. It approaches the problem of a common referential frame between vehicles as a data estimation problem, and that of the optimisation of information exchanges between vehicles as a distributed optimisation problem under constraints. The originality of this work lies in the use of adaptive multi-agent systems (AMAS) to solve them. The AMAS approach is an organisational approach to building complex systems that adapt, continuously and locally, to the dynamics of their environment. It focuses on the interactions between the system and its environment on one hand and between the parts (agents) of the system on the other. These interactions are based on local and cooperative processing of information by the parts of the system, which only have a partial view of their environment. This principle of locality guarantees the emergent nature of the system's operation.

The evaluation of the two modules was carried out using various datasets highlighting disturbances that could affect the system (modification of the environment and intermittence of vehicles in the fleet). The results show that both modules are effective for large-scale problems in a dynamic environment. The use of a local approach to solve the problem avoids an exponential increase in complexity. In the context of optimising the information exchanged, and in order to propose a solution that preserves the confidentiality of the data, the local solution of the problem is not based on the exchange of the CAVs' personal information. Each module has been compared to solutions proposed in the literature for each respective domain (data estimation and distributed constrained optimisation problems). The results show that for the dynamic and large-scale problems considered, the two modules obtain better results than the standard solutions.

Guilhem Marcillaud

IMPLÉMENTATION D'UNE COMMUNICATION COOPÉRATIVE DANS UNE FLOTTE DE VÉHICULES AUTONOMES ET CONNECTÉS

Directeurs de thèse **Marie Pierre Gleizes**, Professeure, Université Toulouse III Paul Sabatier
Valérie Camps, Maître de Conférence, Université Toulouse III Paul Sabatier
Stéphanie Combettes, Maître de Conférence, Université Toulouse III Paul Sabatier

CETTE thèse adresse la problématique de la communication dans le cadre d'une flotte de véhicules autonomes et connectés. Elle s'intègre dans les thématiques des systèmes de transports intelligents et de la ville intelligente. De nombreuses applications ont été développées ces dernières années dans ces thématiques, chacune ayant pour objectif l'amélioration de la qualité de vie des usagers. L'automatisation de la conduite, au centre des préoccupations de l'industrie automobile depuis les années 2000, devrait permettre de réduire le nombre d'accidents, d'améliorer le confort des usagers et également de réduire l'empreinte écologique du trafic routier en général. Elle ouvre aussi la voie à la mise en œuvre de stratégies coopératives efficaces entre véhicules. Comme la coopération repose sur l'échange d'informations pertinentes, il est nécessaire que les véhicules soient capables de connaître quelles informations transmettre et comment.

Cette thèse s'intéresse plus spécifiquement à la communication inter-véhicules ; elle aborde cette problématique comme un système complexe dans lequel évoluent de nombreux véhicules, en interaction les uns avec les autres et ayant chacun leur objectif local. Chaque véhicule perçoit des informations dans son environnement local et sait que parmi ces informations, certaines peuvent être utiles à des véhicules voisins. En tant qu'entité coopérative, il va partager avec son voisinage les informations qu'il suppose utiles. Dans ce travail, une communication inter-véhicules est considérée comme utile si elle vérifie les deux propriétés suivantes : 1) l'information échangée est compréhensible par le véhicule et 2) elle lui apporte de nouvelles connaissances. Dans le cadre d'une flotte de Véhicules Autonomes et Connectés (VAC), ces deux propriétés peuvent ne pas toujours être garanties, en particulier si les véhicules concernés ne partagent pas le même référentiel (par exemple, différentes unités de mesure pour la même information) ou bien si le volume de communications dépasse les capacités du véhicule.

La contribution de cette thèse est double : elle propose un premier module permettant à un véhicule d'adapter une information à son propre système de référence, et un second module permettant d'optimiser les échanges d'informations au sein d'une flotte de VAC. Elle aborde le problème de référentiel commun entre véhicules comme un problème d'estimation de données, et celui de l'optimisation des échanges d'informations entre véhicules comme un problème d'optimisation sous contraintes distribuées. L'originalité de ce travail repose sur l'utilisation des systèmes multi-agents adaptatifs (AMAS) pour les résoudre. L'approche par AMAS est une approche organisationnelle permettant de construire des systèmes complexes qui s'adaptent, en continu et localement, à la dynamique de leur environnement. Elle met l'accent sur les interactions entre le système et son environnement d'une part et entre les parties (agents) du système d'autre part. Ces interactions sont basées sur un traitement local et coopératif des informations par les parties du système, qui n'ont qu'une vue partielle de leur environnement. Ce principe de localité garantit la nature émergente du fonctionnement du système.

L'évaluation des deux modules a été réalisée en utilisant divers jeux de donnés mettant en évidence des perturbations pouvant affecter le système (modification de l'environnement et intermittence des véhicules dans la flotte). Les résultats montrent que les deux modules sont efficaces pour les problèmes à grande échelle et évoluant dans un environnement dynamique. L'utilisation d'une approche locale pour la résolution du problème permet d'éviter une augmentation exponentielle de la complexité. Dans le cadre de l'optimisation des informations échangées, et afin de proposer une solution permettant de préserver la confidentialité des données, la résolution locale du problème ne se base pas sur l'échange d'informations personnelles des VAC.

Chaque module a été comparé aux solutions proposées dans la littérature pour les problèmes d'estimation de données et d'optimisation sous contraintes distribuées. Les résultats montrent que pour les problèmes dynamiques et à large échelle considérés, les deux modules obtiennent un meilleur résultat que les solutions standards.

Remerciements

Voilà la conclusion de trois années de travail, dont quelques-unes en confinement. En regardant en arrière, ces trois années ont été une source riche en discussion, débats, théorie des jeux et de temps en temps du travail.

Je tiens tout d'abord à remercier monsieur René MANDIAU et monsieur Mahdi ZARGAYOUNA pour avoir accepté de relire mon manuscrit de thèse. Vos commentaires et propositions donnent un plus à ce manuscrit. Merci à monsieur Olivier SIMONIN qui a accepté d'être évaluateur et président du jury lors de ma soutenance de thèse.

Je remercie particulièrement tous mes encadrants, officiels ou non qui m'ont accompagné durant ces trois années. Marie-Pierre, ton dynamisme, ta volonté et surtout ta confiance me permettant de réaliser des projets autre que la thèse. Valérie, merci pour ton appui, ton écoute et surtout pour les feux d'artifices. Je te remercie également pour la confiance que tu m'as donnée dans l'enseignement. Stéphanie, merci d'avoir toujours pris du temps lorsqu'il y en avait besoin, même lors des moments compliqués. Merci Elsy pour ta joie de vivre et tes remises en question qui, loin d'être frustrantes, permettent d'avancer. Et un merci à quelqu'un qui m'a déjà reproché (gentiment) de l'avoir remercié.

Je remercie toute les Smackers, petits et grands, jeunes et moins jeunes qui ont permis de faire que ces trois années soient agréables à vivre. Grazie Davide, nos discussions sur la cuisine (d'intérêt purement scientifique) et tes jeux de mots en français sont un souvenir, on va dire mémorable. Un grand merci à Bob, qui le premier m'a fait confiance et a toujours été de bon conseil. Je remercie Kristell qui par sa seule présence met de l'animation dans l'équipe. Ta blague de la clim reste inégalée dans le monde. Augustin, désormais un vieux alors que je t'ai connu jeune, qui a tout fait pour animer l'équipe et dont j'ai tenté de prendre la suite. Merci aux vieux Quentin, Sébastien, Mathilde, JB, Maxime, Walid, Maroun, Florent et je ne remercie pas Valérian comme il se doit.

Evidemment, tout cela n'aurait pas été possible sans l'appui indéfectible de ma famille et des amis. Décompresser est une des choses les plus importantes pour continuer le travail.

Et évidemment, première dans mon cœur, merci à toi Aurore. Merci de m'avoir supporté, surtout dans les derniers moments de la thèse, ta patience est désormais légendaire. Tu m'as accompagné et rassuré tout du long. Ta joie de vivre sont un réconfort de tout temps et je ne peux exprimer par les mots à quel point je t'aime.

Contents

Chapter 1

Introduction

"Going faster", this sentence illustrates a part of humankind evolution. Alone, a human can reach a speed of 45 km/h maximum. Nowadays, with technology, a human can reach the speed of 1227 km/h on earth and even more in the air or in space. It began in 3500 BC with horse domestication allowing to use the speed of another species. Throughout history, with the advancement of technology, came the development of the transportation mode and the infrastructure needed for it to be functional.

Even if some scientists said that it was impossible to safely go faster than 40 km/h, transportation modes were designed and proved them wrong. With the industrial revolution in the 19^{th} century, steam power was handled which leads to the creation of the train and the first car. The car, a mechanic replacement to the horse, was a revolution for personal travel. At first, only available for privileged people, cars became the common transportation mode for many people since its democratisation by Henry Ford.

Vehicles have continued to be improved over the years, getting always faster. A journey that would have taken days to achieve on foot, is only a question of hours now. People were getting closer in time and it has opened the possibility to work in a place further than before. However, with the personal vehicle democratisation came the problem of sharing common space. It was no longer possible to increase the speed with a faster vehicle because the roads were congested. Traffic jams became a daily basis in urban areas and vehicles are stuck at low speed most of the time.

Nowadays, roads infrastructure, safety and the number of vehicles are the limits to the human dream of reducing travel time. It is no longer "How can I have a faster vehicle ?" but "How to use my fast vehicle efficiently ?". Choosing the less crowded roads, optimising my speed and adapting the driving behaviour according to events are the new objectives. The increasing diffusion and accessibility of sensors enabled road traffic to become more "intelligent". Intelligent Transportation Systems (ITS) is a research field that focuses on improving road traffic, especially in large urban areas with dense populations. Improving the road traffic has multiple benefits for both quality of life and economics: pollution reduction [151], travel time reduction [14], comfort [34] and security [143]. It also enables new transport modes and services and those who will benefit the most are those who are unable to drive due to handicap or old age.

1.1 Motivation

With the increased instrumentation of vehicles and the *first fully autonomous vehicle race* that took place in the current years, autonomous vehicles are expected to replace human drivers. Not only they are not susceptible to tiredness and aggressive behaviours but they have the communication asset. It has also a comfort objective allowing the driver to relax instead of driving and allowing non-driver people to use a vehicle on their own.

Using the most advanced means of communication (5G, light communication and protocols) vehicles can share messages and become connected vehicles that can exchange information to extend their understanding of the situation they are in. This open many ways to achieve better coordination for a fleet of Connected and Autonomous Vehicles (CAVs). The objective of communication within a fleet of CAVs is to share knowledge. From this, all connected vehicles have access to improved knowledge, enabling more precise actions. But being connected gives a way for connected vehicles to discuss and negotiate. This is as much efficient as autonomous vehicles because they are able to process many communicated information in a short time. This is the base of many works from the state-of-the-art that present a way to improve one or several road aspects using information exchanges. With it comes the question of "how to communicate efficiently in the fleet of CAVs".

Communication is a basis for many road traffic improvements. In consequence, all these improvements are subject to failure when connected vehicles are not able to properly communicate. So, there is a need to ensure that CAVs are able to communicate. Without going further on the mean used to communicate (networks and protocols), a CAV can have the tools to ensure communication reliability on its own. Indeed, there are two possibilities, coming from the vehicle's design, that may prevent proper communication. The first is the possibility for two different vehicles to not be able to understand each other. The same information is perceived in two different ways between two different vehicles. The second reason is that a connected and autonomous vehicle is not able to process the information in time. In such a large-scale environment as the road traffic (especially in urban areas), if all vehicles communicate everything they gather from their multiple sensors, the amount of communicated information is so huge that no vehicle would be able to process it. These two constraints can be addressed directly by the vehicles, being part of the edge computing domain where computations are distributed at the *edge* of the system instead of using a centralised system [1].

These two presented constraints are the motivation for this thesis work which focuses on solving these two points from the vehicle's side instead of using a centralised system to cope with the road traffic specificities.

1.2 Contribution

This thesis presents two Multi-Agent System (MAS) based modules, respectively LUDA (Learning Usefulness of DAta) and CODCOP (COoperative Distributed Optimisation Problem), providing to a fleet of CAVs the tools to achieve an efficient exchange of information. A MAS is composed of numerous autonomous entities, called agents, interacting with each other in a dynamic environment that they only perceive locally. Focusing on agents, a MAS is highly suitable to model complex systems [122]. As a fleet of CAVs is composed of numerous vehicles in interaction in a continuously evolving world and using sensors to see a part of their world, the parallel between the fleet of CAVs and MAS is natural. Two problems are tackled: 1) the communication understanding and 2) the communication optimisation. The contributions of this thesis are:

- To give an agent the tools to learn to **adapt the referential frame** of other agents' communication into its own, making them usable. By doing so, agents are able to achieve a cross-understanding between them.
- To Achieve a cooperative exchange of communication where each transiting information in the system brings an increased knowledge for receiving agents. This is primordial in the ITS context as many heterogeneous entities interact together increasing the amount of information. Because this amount must be limited, there is a need to **optimise the communications**.

These two problems are addressed while taking into account the road transport system characteristics. The strengths of the proposed modules are:

- **Dynamic**: the road transport system takes place in the real world which is continuously evolving. The solution of a system is only accurate for a short period of time. When the system is disturbed, a new solution needs to be found according to the new state of the world.
- **Open**: the openness of a system refers to the capacity of the system to adapt to the arrival and departure of entities. The disturbance caused does not prevent the system to continue working. While agents are in re-organisation it is still able to provide a sub-optimal solution until the near-optimal one is found.
- **Large-Scale**: in urban areas, the road traffic may reach a huge number. Using a MAS approach with a bottom-up design, the focus is made on the entities behaviour. Doing so, computations are distributed among agents, the CAVs, avoiding the problems of bottleneck inherent of a centralised solution.
- **Privacy friendly**: privacy is a major concern nowadays. More and more techniques are developed using personal data to create information. It is mandatory to respect the passengers' privacy. To find an acceptable solution the CODCOP module does not require the exchange of personal information.

These four characteristics are important in the road traffic context, especially in crowded urban areas, continuously growing. Because communication of many heterogeneous entities is used in multiple applications in the ITS domain and more when including the all smart city context, it is preferred to distribute computation on the entities. In the smart city domain it is now referred as **edge computing** [1]. A local-centred solution allows avoiding problems of the bottleneck of communication that may occur when several thousand entities communicate simultaneously. Doing so, the knowledge is distributed among the entities and local preferences are kept. Private information is stored within the entities and not shared with another system without knowing what happens to it.

1.3 Manuscript Organisation

This thesis consists of five more chapters as follows:

- **Chapter 2 Intelligent Transport System Context.** This chapter provides an overview of the Intelligent Transport System domain, the benefits that are expected to be brought, and all the different components composing it. Then the concept of CAV is detailed with the different levels of autonomy to understand what is expected from it. After that, a state-of-the-art of different applications for the fleet of CAVs is made. All of them use communication and the exchange of information to improve at least one road traffic characteristic. Communication is indeed primordial for enabling cooperative strategies between CAVs. They need to be able to understand each other, achieving a **cross-understanding** and not creating too much noise by **optimising their communication**.
- **Chapter 3 Methods and techniques.** In this chapter, a state-of-the-art of the techniques that can be used to address one of the two problems is given. **Cross-understanding** and **Communication Optimisation** belong to two known classes in the scientific literature. The first one can be addressed as a **Data Estimation** problem while the other belongs to the **Distributed Constraint Optimisation Problems**. Both domains give a guideline of what is the best kind of solution to address a problem with the characteristics of dynamicity, openness, large-scale and privacy. From this, the Adaptive Multi-Agent System (AMAS) approach is presented as it is efficient towards the stated challenges of **dynamicity, openness, large-scale and privacy**.
- **Chapter 4 Contribution.** In this chapter the architecture of the contribution is detailed. It presents two different AMAS, each focusing on one challenge (Cross-understanding or communication optimisation).

The first one called LUDA provides an agent with the tools to adapt an information from another referential frame into its own in a large-scale environment. The second called CODCOP focuses on a cooperative strategy for a fleet of CAVs to exchange information with two constraints 1) not overloading an agent with too much communication preventing it to process all of it and 2) only the most useful information are exchanged.

- **Chapter 5 Experiments.** Each module is evaluated to assess its efficiency confronted to the required characteristic (dynamicity, openness and large-scale). First, the experimental conditions are presented. Comparison with standard tools is made whenever possible. Comparisons assess the efficiency of proposed solutions in terms of tools. Then an exploration of each characteristic is made to observe the solutions' behaviour with dynamic changes, vehicles arriving/leaving and the scale growing. Moreover, the proposed solution does not exchange private information to find a solution.
- **Chapter 6 Conclusions and Perspectives.** This thesis contribution has focused on the ITS context using an AMAS approach to tackle problems in two different domains. In this chapter, a conclusion is given for every domain/context and the perspectives of use and improvements are presented.

According to the given outline, the next chapter presents the Intelligent Transport System (ITS) context characteristics and purposes.

Chapter 2

Intelligent Transport System and Connected vehicles

Objectives of this chapter:

- Presenting the domain of Intelligent Transport Systems and the benefits it is expected to bring. Precisions about the fleet of CAVs concept as well as the means of communication of all involved intelligent entities are then given, using a network and a dedicated protocol. Then a definition of what is a fully "Autonomous Vehicle", what are the main components and the requirements to authorise them on public roads is given.
- Presenting works from the scientific literature that take advantage of the fleet of connected and autonomous vehicles to improve road traffic.
- Studying the challenges of cross-understanding and communication optimisation within a fleet of CAVs.

2.1 Purposes of Intelligent Transportation System

Through the last decades, urban areas have attracted more and more people from rural areas inducing a concentration of population in urban areas. In 2020, 12 megalopolises have reached a population of over 20 million up to 38 million for Tokyo in Japan [26]. The limited space available induces an ever-increasing extension of the city and an increase of the travel distance for the population to reach the town centre. Generally, workplaces and big cultural infrastructures are downtown attracting a large population at the same time. Contrary to the population evolution, the transportation infrastructures evolve much less quickly leading to daily traffic jams in large urban areas. The road size is no longer sufficient to support the number of vehicles, and they have a weak perspective of improvement due to the lack of available places. This increasing number of cars is harmful for the population and more globally for the planet. In 2004, road traffic is responsible for 13% of CO_2 emissions in the world. This issue is even more important in large urban areas with the concentration of pollution-induced by traffic jams. As a consequence of this pollution, the number of asthmatics and allergy sufferers increases every year, particularly near cities. Furthermore, traffic jams are not only negative for health but also the economy. Inhabitants of large urban areas commonly spend more than

Figure 2.1: The driving cost : *The Economist from INRIX*

one week in traffic jams in one single year [68]. The cost of traffic jams is evaluated at several billions of euros. For example, in 2013, France lost 13 billion euros, and this number is expected to grow until 2030 [67]. Figure 2.1 presents the time lost and the cost of congestion in large cities. In the most crowded areas, a driver may lose more than 100 hours each year. This loss has a big impact on big megalopolises because each lost hour cost more. For example in the figure 2.1, with almost the same cost by driver between New York and Los Angeles, the total difference is 14 billion. Addressing road traffic congestion is not only a comfort objective but also an economic one that has to be improved.

Another major problem with road traffic is its responsibility for 1.25 million deaths each year in average (1.35 million in 2016 as seen in table 2.1) and an even greater number of injuries resulting in life-long disabilities. The international campaign ”Make Road Safe” tells that every 3 minutes in the world a child dies [40]. Road traffic is the first cause of mortality for young people in USA and Europe. During the last decades, governments have introduced several laws reducing the number of accidents but Europe still has 80 000 deaths each year as seen in the table 2.1.

The majority of road accidents, 90%, are due to human errors such as inattention or drunk driving [43]. Incompetence and aggressive driving are the principal causes of traffic jams. The majority of drivers do not respect the speed limit and/or the safety distance, responsible for a high number of accidents.

2.2 ”Intelligence” in ITS

”Smart” technologies are integrated into more and more domains. From the development of Smart cities to the connected coffee machine, those technologies are meant to improve the quality of life for the users. The transportation system is not an exception and since 1970 Intelligent Transport Systems (ITS) have been evoked as an ecosystem for the improvement of road traffic [83, 119]. ITS includes all smart technologies that

Country/region	Road fatalities per 100,000 inhabitants per year	Road fatalities per 100,000 motor vehicles	Total fatalities latest year (adjusted/estimated figures by WHO report)	Year, data source
World	18.2		1,350,000	2016
Africa	26.6	574	246,719	2016
Eastern Mediterranean	18.0	139	122,730	2016
Western Pacific	16.9	69	328,591	2016
South-east Asia	20.7	101	316,080	2016
Americas	15.6	33	153,789	2016
Europe	9.3	19	85,629	2016

Table 2.1: Road fatalities in 2016

help to monitor, control and improve road traffic [17] and quality of life for road users. ITS includes many different domains (buildings, mobility, services,...) and this thesis focuses on the domain of Intelligent and Communicating Vehicles.

A Connected and Autonomous Vehicle (CAV) is an entity composed of many different components mechanical as well as software, all with much room for improvement. No miracle solution solves everything but many components working together tend to find a preferable solution. Each one of these CAVs, having its purpose, can be improved to increase security and comfort for the passengers. A CAV may be associated with other CAVs, thus forming a fleet. To be considered as a fleet and not an organisation of self-absorbed entities, CAV is required to have social skills. These skills can range from selfish to cooperative to altruistic. Cooperation is a way to improve many road traffic aspects allowing better coordination and an improved understanding of the situation [2]. Characteristics of the intelligent vehicle are presented hereafter. First, a short introduction about the hardware embedded in an intelligent vehicle is proposed. After, all levels of autonomy of a vehicle are characterised to know what level is taken into account.

2.2.1 The Hardware in an Intelligent Vehicle

Car constructors innovate more and more each year. The gap between the Ford T (one of the first mass-production vehicles first built in 1908) and a recent car is huge. For many decades, car constructors have been improving cars to make them faster, safer, more comfortable, and more attractive for consumers. In recent years, more and more constructors have begun to work on ”Intelligent Vehicles”. From a mechanic and hydraulic machine, the car is now massively composed of electronic parts. More and more sensors are embedded in vehicles, gathering a massive amount of road traffic data (speed, road state, vehicles position, etc). From this data, the vehicle is now able ”to see” the world around it. This adds intelligence to understand the information and make decisions: the vehicle gains in decision-making autonomy.

Each intelligent vehicle is equipped with an On-Board Unit (OBU) which is in charge of (i) reading data from on-board sensors, (ii) determining which manoeuvres to initiate, and (iii) exchanging sensors data with neighbouring vehicles [121].

2.2.2 Towards a Self-Driving Car

The purpose of a self-driving car is to assist or even replace the driver of a vehicle. It is seen as a mean to improve the comfort of the user (not requiring the user’s action to operate the vehicle) but also a mean to improve the overall traffic by lowering the ”bad” behaviours of human drivers. The Society of Automotive

Engineers has defined in 2014, 6 levels of automation that enable to classify a vehicle type according to its capacities:

- **Autonomy level 0**: the vehicle does not assist the driver at all.
- **Autonomy level 1**: the vehicle assists the drive on specific roads. This level was first achieved in 1939 with cruise control. The vehicle can regulate the speed or the direction, limiting bad actions from the driver. The driver is still in charge of the driving.
- **Autonomy level 2**: the partial automation vehicle can take control of the driving in a specific context. In 1995, Bosch sold a lane correcting system able to make the vehicle correct its trajectory alone to stay between the line. The driver is still mandatory to look after the vehicle action. In addition, the vehicle can park itself without any help from the driver. This is still the autonomous level of current cars in 2020.
- **Autonomy level 3**: this is the first level of limited automation. The vehicle can drive safely on most roads without any help. However, a human driver is still mandatory because of specific situations that the vehicle is incapable to handle. For example, a vehicle can drive daily except in snow weather, and on difficult roads like mountains or if the traffic is too dense. It is important to note that the human is not required to watch over the vehicle but must be able to take back control in a short time if necessary. Current researches on this subject are studying how to give back the control of the vehicle to the human driver in a safe manner [7].
- **Autonomy level 4**: complete automation but on specific conditions. The major difference with level 3 is the possibility for the vehicle to handle problems it encounters. Where a level 3 vehicle must give the control back to the driver with traffic densification, a level 4 vehicle can continue to drive safely. The only exceptions remaining concern specific roads like mountains or exceptional climatic conditions like snow.
- **Autonomy level 5**: the last level removes all the possibilities for the human to drive: the vehicle can drive safely and efficiently on every road in every weather condition.

Vehicles equipped with Autonomous Driving System (ADS) levels 3 to 5 [139] are useful in preventing accidents related to human errors. However, there is a sociological challenge about the acceptability by a human of such vehicles [100]. Not everyone is willing to leave the driving to a machine because they like to drive or because they lack confidence in it. Moreover, leaving the decision to a machine raises ethical problems in some specific situations where the only answers possible result in a human death. The Moral Machine platform of MIT collected responses from people around the world and no consensus was found [98]. However, in the end, a decision must be taken and the possible responsible parties (manufacturer, owner, ...) must still be determined.

ITS does not only refer to intelligent vehicles but also intelligent road infrastructure. All over the years, urban planners improve road infrastructures to adapt to an ever-increasing number of vehicles. However, building new infrastructures along the roads is complex, costly, and environmentally damaging. Recently, urban planners have taken advantage of advances in technology to integrate smart infrastructures with existing ones. Some current examples are: traffic lights that adapt their colours to improve traffic, sensors that detect the beginning of traffic jams and barriers open autonomously, etc.

In the next sections, Connected Vehicles (CVs) and Autonomous Vehicles (AVs) are studied individually. Section 2.3 describes how a vehicle is considered as *connected* what are the means used to interact with other CVs. Then, section 2.4 gives an overview of how a vehicle can be considered as *fully autonomous* and

the components. After, both characteristics are grouped to study the Connected and Autonomous Vehicle (CAV). Finally, a state-of-the-art of works that takes advantage of a fleet of CAVs is proposed. It highlights the requirements of such an application.

2.3 Connected Vehicles

A connected vehicle is a vehicle with an on-board unit (OBU) able to connect to a network and communicate through it. It is important to understand what are the characteristic of the network and the protocols used. In this section, the different types of communications that can take place in a fleet of CAVs are first presented. Then, a study of the dedicated network, Vehicular Ad Hoc Network (VANET) is given. Following it the different protocols used are detailed, the motivation of using the broadcast protocol is justified. A new means of communication using light is presented and communication challenges are extracted. A discussion about the domain of the connected vehicle is finally given.

Currently, there are only a few ways to share any information with another driver on the road: the car indicator, the horn, the lights and the human arm. Recent advancements in network technologies enable vehicles to send specific messages to other vehicles and road infrastructures [138]. It opens the way to the exchange of more messages with precise content between CVs. In the ITS context, four communication categories are discussed [90] and illustrated with the figure 2.2:

- Vehicle-to-Vehicle (V2V) Communication (orange arrow in the figure 2.2): the communication is established between two vehicles. V2V has the major advantage of not requiring any network infrastructure [121]. It is highly useful in rural areas and less developed cities. This communication is particularly efficient for the coordination of truck platoon [81] (platooning is a way to group vehicles into a coalition facilitating cooperation inside the coalition).

- Vehicle-to-Infrastructure (V2I) (blue arrow in the figure 2.2): the communication takes place between a vehicle and an intelligent infrastructure. An intelligent infrastructure can collect data from the vehicle and/or send data to the vehicle [77]. A well-located intelligent infrastructure is likely to have more information about the current state of the traffic. Through these communications, it can help an Autonomous Vehicle (AV) to take its decisions or even coordinate several vehicles.

- Infrastructure-to-Infrastructure (I2I) (green arrow in the figure 2.2): for example, two intelligent infrastructures exchange useful information to coordinate traffic lights to relieve road congestion on a specific road by forcing traffic lights to become green.

- Communication between Vehicle and Pedestrian (V2P): the vehicle is connected to the smartphone of the pedestrian to discuss with him. But, as V2V communication, the range is not high. Another means is to equip the vehicle with an external display that indicates to pedestrians the AV decision. One simple example already existing is blinkering.

2.3.1 Network

Talking about a fleet of connected vehicles is now possible thanks to network progress. The ground-breaking innovation in the network domain is the fifth generation (5G) mobile communication networks [55]. It is expected to be highly scalable, to provide ultra-high reliability, low-latency, high throughput, flexible mobility, and energy efficiency [12]. Internet of Things (IoT) is a concept that represents the interconnection of many intelligent devices. When investigating specifically connected vehicles, this concept becomes the Internet

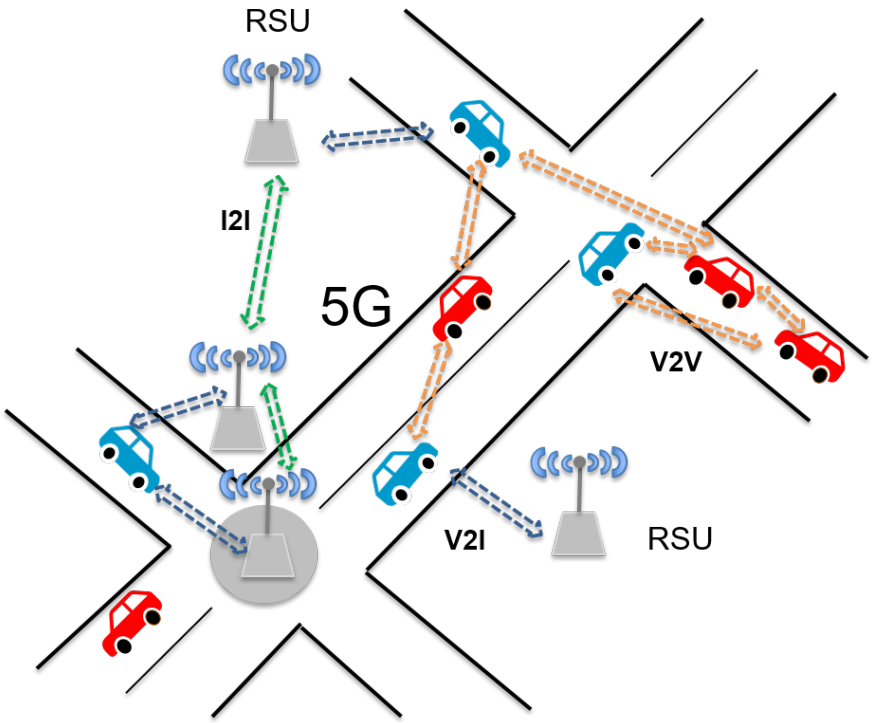


Figure 2.2: An example of communication in ITS with eight CVs and four Road Side Unit (RSU). Using a 5G network, entities are able to exchange information. Orange arrows represent V2V communication, blue arrows represent V2I communication and green arrows model I2I communication.

of Vehicles (IoV), the interconnection of many intelligent vehicles. The IoV is possible thanks to the ITS infrastructure and the specific network. As the IoV has its particularity strongly induced by the dynamics, networks challenges have been raised. Specific networks and protocols have been proposed and extended to tackle challenges.

To communicate, vehicles need to be equipped with communication devices and use specific network protocols. Protocols have evolved from those of Mobile Ad Hoc Network (MANET) to those currently used in IoV, Vehicular Ad Hoc Network (VANET). A succinct description of each one of them is given hereafter. Note that this description stays at a high level as it is not in the thesis scope to discuss networks; for more information, readers can have a look at *Ad Hoc* networks books [111], [140] and [87].

These network solutions are some basis that enable many works to exist. In recent years, numerous works taking advantage of networks improvement have arisen. The network is a domain still in active research and although 5G is only being deployed, research institutions began to develop the 6G [66] and start to look after eco-network.

2.3.2 Mobile Ad Hoc Network (MANET)

The network is a wide research domain still expanding and has many applications cases, including the road network. By looking precisely at the road transportation system two protocols stand out. They have been designed according to the application domain characteristics. Indeed, the road transportation system includes by nature highly mobile actors. In addition, the road structure is not obstacle-free for communication. Walls are the main concern when sending a signal which is degraded because of them. This limits heavily the range and reliability of communication. Before talking about a specific vehicular network, a network used for smart city purposes is described.

The MANET is a wireless, multi-hop, self-configuring network used in the IoT domain [21]. It has been extended to several different networks having the following features: dynamic topology, variable capacity links, energy-constrained operation, and limited physical security. Some MANETs are generally restricted to a local area of wireless devices, while others may be connected to the Internet [87]. The dynamic topology of this network has enabled to address transportation applications. This characteristic has opened the way to the development of a specific protocol for vehicular communication.

2.3.3 The Vehicular Ad Hoc Network (VANET)

The Vehicular Ad Hoc Network (VANET) is a special class extended from MANET with predefined roads. It relies on specific authorities for registration and management, RoadSide Units (RSUs), and On-Board Units (OBUs). RSUs are widespread on the road edges to fulfil specific services (like controlling a traffic light) and OBUs are installed in the vehicles navigating in VANET. All vehicles are moving freely on the road network and communicate with each other or with RSUs and specific authorities [59]. Vehicles equipped with an OBU can be considered as a node in the network. Vehicles situated in communication range create a global network, thus becoming a fleet of connected vehicles. Many VANET protocols have been designed. The three majors ones used are presented hereafter, the Unicast Protocol, the Broadcast Protocol and the Multicast/Geocast Protocol [76].

Unicast protocol

This protocol uses a multi-hop strategy to transfer information from a source node to a destination node [38]. This protocol has been much used in VANET as it takes advantage of the high traffic density to transfer information. However, the use of multi-hop causes latency to the information. If numerous hops are required

to reach the objective vehicle, it may take more time and thus increase the possibility of being outdated. The unicast protocol is mainly used in platoon formation because of the fast exchanges it provides, but this is not the unique fleet form addressed here. Every type of fleet of CAVs is considered including a fleet with vehicles having opposed goals.

Broadcast protocol

This protocol can send a message to every other vehicle in its range [149]. It is efficient for the dissemination of information regardless of the receivers. However, there is no means to prevent a vehicle to receive it. Disseminating information is effective when the sender believes that this information is worth sharing, even if it does not know vehicles around him.

Multicast/Geocast protocols

It allows performing data communication from a vehicle to multiple **targeted** vehicles [132]. The geocast extension targets a geographical area instead of specific nodes [9]. It is mainly used to disseminate information with an emphasis on which vehicles receive it (for example the vehicle behind) and to improve message security (selecting a subset of the receiver). Moreover, it enables to target the information to send according to the information content thus efficient for context-dependent information. For example, the state of an intersection is useful to vehicles arriving at this intersection but not to those that have passed through it. However, this requires the vehicle to have a correct understanding of its environment.

2.3.4 Light communication

In recent years, another communication means has emerged using lights [167]: it is called Visual Light Communication (VLC). Taking advantage of the light speed, this technology also helps to prevent hackers to use conventional communications ways like radio and the internet to disrupt communication. The security relies on the sender's position to check the source. Infrastructures can also use this technology. Vieira et al. [154] proposed to equip street lamps with VLC to control the traffic inside an intersection. However, this has appeared recently and the application has not yet been proved. Obstacles may be more problematic for light communication than for conventional communication methods (for example, the light is completely blocked by walls).

2.3.5 Challenges for Communication in IoV

As stated by Bwalya et al. [22], there are many challenges for a vehicular network that need to be addressed. Among these challenges, three are extracted to conduct this thesis work.

- Usually vehicular networks are based on fixed or mobile nodes and the infrastructure. As mobility aspects get better in VANETs, a mobile vehicle must be able to communicate using either mobile infrastructure embedded in GPS or the vehicle itself. The **mobility characteristic** is the main challenge that constraints a lot by adding a new dimension to the problem.
- The nodes in the V2V communication network scenarios are highly mobile presenting, all constraints involved like signal propagation and persistence. There is a need to **design dynamic protocols** that can evolve accordingly with the network states, and thus maintaining reliable communication. This also has an ecological impact because protocols and nodes should not be constantly used if not necessary, thus lowering the energy cost.

- Masini et al. [93] state that a fully connected car will require a massive amount of computing power and super-high-speed communication systems, with latency lower than 5 ms and reliability higher than 99.9%. In consequence, it is mandatory to adapt communication to the network characteristics, limiting what could impact **latency or reliability**.

The first two challenges highlight the characteristic of **dynamics and openness** of a system into a fleet. The network is constantly evolving and the actors' mobility increase is another obstacle to take into account. Resources are limited and considering a CAV as a critical system makes it subject to critical decisions that cannot be postponed (justified because its decision can result in a human casuality). **Communication may need to be limited** to respect the maximum latency authorised. Because many traffic improvement solutions use an exchange of information to enable a strategy (for example, the coordination inside an intersection), a fleet of CAVs needs to find an equilibrium between exchanging required information and avoiding system congestion.

2.3.6 Applications of Connected Vehicles

With the communication advancements, has come new forms of mobility for urban areas. In recent years, the concept of *Mobility as a Service* (MaaS) has emerged. It is based on a new vision where a fleet of CAVs replaces private vehicles. Instead of using its private vehicle, it is possible to command one available through an application. The benefits are expected to be multiple, lowering the number of vehicles on the roads, lowering the required parking slots, and enabling cooperative routing strategies. Daoud et al. [28] have proposed a generic model for online on-demand transport. Their solution supports different allocation mechanisms processing and considers autonomous vehicles that communicate via peer-to-peer radio channels to meet passenger's requirements and satisfy trip requests. This study focuses on the driving problematic but that is not all the applications spectrum. Many works focus on cooperative strategies to find a parking spot quicker [165]. Using the VANET protocol, vehicles cooperate by exchanging information about free spots. By doing so, the overall time-wasting is reduced.

2.3.7 Discussion

The theme discussed in this thesis is the cooperation between CAVs. Assuming the existence of a network with the quality of the 5G network and the VANET protocol, enables to consider a fleet of CAVs as achievable. Indeed, CAVs can exchange information fast enough to negotiate and coordinate. Light communication is interesting progress but needs more research on the feasibility. Moreover, works using it present a limitation on the number of different information that can be shared.

Challenges in vehicular communication are related to the specificity of the domain. The high dynamic part is the most challenging characteristic that needs to be addressed to make communication reliable and useful. Networks are still evolving, making it possible to bear a still increasing amount of data. Likewise, some information can be heavier with sensors' improvements. Many applications and cooperative strategies based on the exchange of information have arisen which are more discussed in section 2.6. After the connected part, the second part of a CAV is the **autonomous** one, the next section studies the Autonomous Vehicle (AV) domain and its characteristics.

2.4 The Autonomous Vehicle (AV)

An autonomous vehicle is a robotic entity with many components from simple bolts to advanced computers. With recent advancements in both Artificial Intelligence and robotic control, it is now possible to provide a *brain* to the vehicle. The objective is to replace the human driver notably to improve the driving and road traffic quality (security, pollution, and travel time).

A short introduction on the subject is given in the next sections. This section is constructed with the following plan: the requirements for an AV to be accepted on public roads are first identified then the common behaviour of an AV is described.

2.4.1 Towards an AV

Driving a vehicle is a complex task requiring a human to take into account the vehicle to handle, his objective to reach, the road as well as all the road users to observe. However, even with a driving license, the human driver is far away from being perfect on the road. Studies have shown that nearly 90% of accidents are due to human errors, aggressive driving or negligence [11]. Moreover, a human is subject to emotions (stress, disease, tiredness, anger, and other negative mental states), which may impact negatively driving. The human is not a perfect driver, but he is allowed to drive on public roads despite its flaws. A fully autonomous AV is expected to match three characteristics to be accepted on public roads:

- It is better than a human driver. Its integration on the road traffic must be an improvement overall for safety and comfort.
- It must not be worse than a human driver in any scenario. Even if the AVs can significantly reduce the number of accidents, each accident implying an AV calls the use of such vehicles into question.
- It must be understood by road users and not be a mystery that makes people uncomfortable. AVs must be accepted by human road users. For example, a pedestrian will seek the driver's eyes before crossing the road to be sure that he has been seen. With no driver, pedestrians may feel unsafe to cross.

An AV must drive at least as well as a human driver but not less than a human driver. The first step for designing an AV is to make it able to replace a human driver being in a perfect state of mind but augmented with better computations. This driving capability has been called the "Driving brain" of the CAV by Loke [85].

2.4.2 Driving Brain of a CAV

This is the core layer for an autonomous vehicle. It can take decisions to safely drive and must be self-sufficient according to autonomy level. Depending on its capacity, this driving brain has a particular degree of autonomy (see autonomous level earlier). Currently, constructors are mainly focused on improving it to achieve a level 4 autonomy degree. When writing this thesis, the higher degree of autonomy achieved is 3 for vehicles driving on common roads.

An AV is expected to be a better driver than a human driver as it is not subject to emotions (tiredness, aggressive behaviour, or inattention). However, it can be subject to errors (or failure) like any other system. A real road traffic improvement on the real transportation system is yet to be proven efficient and observed. Most of the reluctance to the successful introduction of AV in public roads comes from two major reasons: 1) hardware failures and 2) difficulty to prove that a system is efficient in every case. The introduction of such vehicles will come progressively under human supervision until their reliability is proven for any situation that may happen.

The driving brain of an autonomous vehicle is composed of several modules, each responsible for a specific task allowing the system to efficiently control the vehicle. As many components may be involved and each constructor using its own, three parts stand out. Forming a cycle: observe, understand, and act in the world.

Observe the World

A driving brain requires to have information about the world it is evolving within. It takes decisions by a reasoning process using the knowledge of its environment and the objective it has to reach. As a human driver, an AV needs to *see* the environment in which it evolves. Thus, the vehicle's *eyes* are the embedded sensors (like LIDAR, camera, radars speed, position, obstacle detection ...) that provide the driving brain information about its environment [120]. It perceives two types of information: 1) the vehicle internal state information called **endogenous** and 2) the environmental information (around the vehicle) called **exogeneous**. All these information, collected thanks to the many sensors, are stored in a knowledge base. With the combination of both information, the AV can have an accurate representation of the world around it.

However, those observations are subject to sensor failures. Adding more and more sensors increases the number of available data. But energetic consumption and data analysis are a challenge, especially with electric vehicles. The access speed and the volatility of information are also important. A compromise must be found between the number of collected data number and their analysis (time and quantity)

Understand the World

Having information is sometimes not sufficient to understand it. For example, having an image is not useful if no information can be extracted from it. Image treatment (through various algorithms) enables the detection of eventual obstacles on the image, and so project it into the environment. The AV's brain must be able to transform data into knowledge used by a higher-level process, for example, an obstacle detection algorithm [145], used to extract several objects from the image as obstacles positioned in the referential frame of the vehicle. Thanks to it, the vehicle can understand if an obstacle is on its trajectory. Many techniques are constantly improved to extract useful information from raw data. The most common methods are deep learning techniques. Such methods require a great amount of data and a proper training time [146].

Act in the World

This is the last part of the AV's decision cycle. It represents the actual effect a vehicle has on its internal state and the environment. A vehicle performs actions through its actuators (accelerator, brake, direction, etc). An advantage of using an autonomous decision process is that it computes with precision the best value an actuator must achieve. However, there is a gap between the action decision and the actual effect. Current vehicles are already embedded with helping control devices, like assisted direction or turbo. All those devices have been proven to improve the driving experience [56]. Advancements in robotics control enable an increase in control precision to reduce the gap between the calculated value and the real value. Vehicle control is still an active research field to counter unwanted effects [155] caused by noise or errors.

2.5 The Social Connected and Autonomous Vehicle

Instead of relying only on its own perceived information, an AV is now part of a network connecting many vehicles, called the Internet of Vehicle (IoV) [163]. IoV is similar to the concept of the Internet of Things (IoT) but adapted to vehicles. Each vehicle is a node and is connected to other nodes to exchange information.

2.5.1 A Social Individual

The Cambridge dictionary defines the term social as *relating to society and living together in an organised way*. From this definition three parts are important. Society refers to all individuals evolving within the same area, each with his desire. These individuals are aware that they are not alone and they can interact which each other. Interactions follow a code known by both individuals. Using a different code may lead to difficulties for interactions. The last part, "organised way", signifies the importance for an individual to act in accordance to their own and other desires, leading thus to a collectively functional society.

Connected and Autonomous Vehicles can be seen as indeed social individuals. As an autonomous entity, it behaves without supervision from any controller. Doing so, already existing social individuals are forced to cohabit with new ones. To be accepted it is required that they follow the existing social code [4]. That means, being able to interact with people but also using codes that are understood by other connected entities. The human world and the CAV world have to cohabit harmoniously and efficiently.

2.5.2 The Social Brain of a Connected and Autonomous Vehicle (CAV)

A Connected and Autonomous Vehicle is considered a social entity evolving with other social entities (CAVs, connected infrastructure, pedestrians,...). Following the Cambridge definition, a common code must be shared to make interactions possibles. Loke [85] introduced the concept of the "Social brain" of the CAV as a second layer in addition to the driving one. He highlights the necessity for CAV to perform social interactions and social signalling. The range of potential interlocutors is large including vehicles of course but also other road users: pedestrians, motorcycles, cyclists, electric scooters, and if we think further, robots. Let's not forget that CAV is likely to be evolving within smart cities. Thus, they are likely to interact with all IoT services disseminated all around the roads facilities.

The social brain is then a dedicated part of a CAV for social interactions. Among all components composing a CAV, this is the one this thesis focuses on, in particular, the understanding part between connected entities. Interaction with non-connected entities like pedestrians is a domain closer to sociology and ergonomy and it is not addressed in this thesis work.

2.6 Motivations for a Cooperative Fleet of CAVs

Although many fleet of CAVss have been developed in recent years, it is difficult to obtain an artificial system that can efficiently address all situations encountered during a trip. This problem is increased by the necessary use of simulators to test these fleet of CAVss. As road traffic encompasses many different situations, it is then very complicated to find a simulator able to model all encountered situations including a large number of vehicles, with characteristics close to reality [142].

This section introduces briefly the fleet of CAVs and then focuses on the cooperative fleet. This study is not exhaustive as the number of existing works is huge but gives an idea of the large spectrum of this subject. It studies existing works from the scientific literature on a cooperative fleet of CAVs and has three objectives. The first objective is to provide the reader with more insight on how cooperation improve road traffic with a fleet of CAVs. The second one is to extract how communication and network help; the assumptions, the limits and the requirements necessary for a solution to function properly are presented. The last objective is to study the parameters of the context that influence cooperation: the number of CAVs, the situation complexity, and the fleet structure. From this analysis, challenges have been extracted and motivate this thesis work.

2.7 Fleet of CAVs in Literature

The amount of existing works about Intelligent Transport System is huge. This domain is wide enough to include multiple disciplines from the research of the shorter path to micro situation improvements. Here a focus is made on the artificial intelligence domain and the strategies of cooperation. Most of the works analysed are fleet of CAVs only but a few of them are linked to intelligent infrastructure strategies. A fleet of CAVs is likely to be confronted with a situation where an intelligent infrastructure manages a specific aspect of a situation. Thus, the fleet is in interaction with this intelligent infrastructure.

As a driver, a human or an Autonomous Driving System (ADS) has to ensure the safety of all its passengers. This point is assumed as covered in all the studied works. This study focuses on how communication (What kind of communication and its characteristics) is used to improve the overall road traffic state. Presented works are selected because they match one of the following criteria: 1) it shows the use of communication to improve a transport characteristic successfully using particular information; 2) it gives an insight on the encountered specific road situation ; 3) the used strategy is efficient and interesting.

The comparison is based on the situations addressed in the studied systems. Multiple characteristics such as the purpose of the system, the traffic mix, and the volume of communications are used to study them.

During a trip, a vehicle can encounter many different situations such as (i) **traffic congestion**, (ii) **intersections**, and (iii) **lane merging**. Whatever the encountered situation, a vehicle has to make the best decision in coordination with other vehicles, to manage the situation optimally and safely. Several examples of scenarios simulating such situations are detailed in the next section.

2.7.1 Comparison Criteria

Eight relevant criteria are identified in the literature to highlight the strengths and weaknesses of the studied systems. They enable to characterise and give an objective comparison between the proposed solutions. From these criteria, it is possible to identify the *ideal* fleet of CAVs and what is required from such a fleet, thus establishing a context for this thesis.

1. **System objectives**: Among the compared systems, three main purposes stand out: minimising travel time [6], reducing energy consumption and pollution [118], and smooth driving [151]. It should be noted that these purposes are not independent. For example, by minimising the travel time, the pollution can be reduced. Reducing the distance will reduce pollution while increasing a vehicle's speed also increases its gas consumption. This criterion is used to measure the benefit to use a given strategy and communication.

2. **Decision locality** [14] : being in the context of a fleet of CAVs, the implementation of the control process, either distributed or centralised, is studied. Apart from being a choice, this criteria shows its utility on overall communication. If the CAV's control is centralised (using a superior controller), it requires a communication flux with the controller. This cause an increase in the communication volume. Indeed, in addition to the information communication, all being sent to the controller, commands for each CAV are also sent through the same network. Other drawbacks are still to be considered: computation time and energy consumption for the controller. Whereas, a distributed solution leaves the decision process to the CAV, so no command is sent through the network. This is important for a critical system where a bottleneck of communication may cause human casualities.

3. **Communication volume** [138]: connected vehicles can receive and provide data through increasingly efficient communication. The amount of data flow can become very large and thus be an issue. With no standard for any information, the form of the communication depends on the state and rate. A huge

volume of communication can overload a CAV which would be unable to process everything. A strategy requiring too much communication would be difficult to be used in parallel with other strategies.

4. **System robustness** [107]: Robust system can react to unexpected events. Two main solutions are available to assess the robustness of CAVs in the event of disruptions: 1) ensuring that all consequences can be predicted in advance or 2) being able to react in time.

5. **Knowledge of the scenario**: the experimentation of a system takes place within the framework of a specific scenario that CAVs might know in advance. This knowledge influences the reaction of a CAV when it detects unexpected events in the scenarios. To keep the solution as generic as possible and not context-dependent, scenario information should be hidden from the CAVs.

6. **Mixed traffic** [123]: the presence of non-autonomous vehicles and other types of users (pedestrians, cyclists, ...) must be taken into account. A strategy based on the full cooperation of all involved entities can only be used in a fully autonomous scenario where all entities agree to cooperate.

7. **Scalability**: the number of vehicles influences the reliability of the system and may reveal specific problems that need to be considered. Scalability is a characteristic to look after as limits concerning the overall amount of CAVs and information cannot be predicted. Scalable solutions are more likely to not be limited by future advances in hardware and society.

8. **Simulation model** [142]: to know its efficiency, a system is evaluated under certain conditions. Microscopic models are used to observe a specific situation with a few numbers of vehicles with a representation close to the vehicles' dynamics. Macroscopic models on the other hand aim at observing the flow of a large number of vehicles (the traffic in a big city).

The following sections present and evaluate Intelligent Transport Systems identified in the literature that propose a solution to a road problem. The purpose of this review is to extract a guideline and the requirements for a system able to manage all road situations (or at least several ones). To have a better understanding of how communication is used and its benefit. Each system is studied according to the eight criteria defined. The study is organised according to the road situation the system addresses. The number of existing different road situations or that may happen in the future can be considered infinite. However, according to the literature, these situations are divided into 3 classes [2]. The first one considers a situation where the overall **traffic flow** needs to be improved: **traffic congestion**. Such a situation is represented by one to several roads, depending on the model. With several roads, a path-finding strategy can be taken, which is not possible with only one. For both, the objective is to reduce the travel time for each user. Depending on what is wanted, the objective can be to decrease the total travel time or to decrease the worst travel time in the system. The second class includes all road situations with **an intersection**, which is the main issue to solve. The last class is the **lane merging** one, an issue addressed in many works.

2.7.2 Panorama of Fleet of CAVs for Traffic Congestion

The objective of these fleets of CAVs is to maintain smooth traffic. Usually, such road situations are in an urban context (big cities), involving a high number of vehicles causing a slowdown or even a traffic jam. Two axes, that aim at improving the overall traffic flow, are investigated in this section: first how a fleet of CAVs can choose a path that reduces the congestion, and second, how to smooth the traffic with better driving behaviour.

Learning the Best Path Based on the Road Congestion

In large urban areas, congestion problems on main roads can be mitigated by the use of CAVs and learning mechanisms according to Barthelemy et al. [14]. The authors present a Multi-Agent System (MAS) with CAV agents. Some CAV agents, called strategic, choose the best path to improve travel time using a neural network. Each agent uses its knowledge on nearby roads to take its decision. The experiment focuses on the Chicago city network, by varying the proportion of strategic CAV agents in the system. The authors observe that an increase of the strategic CAV agents also improves the traffic flow. Using **only local data** causes a low volume of communications and agents do not need to know the scenario to learn and to choose the best route. In this experiment, Matlab is used to simulate at the **macroscopic level** a large number of vehicles and therefore to **successfully scale up**.

The use of learning mechanisms provides great results on the specific situation, here the traffic at the urban scale. The drawback is that every different scenario requires a CAV to learn again with the new scenario characteristics. Moreover, unexpected events (accidents, roadwork, etc) events may lead the learning to be inadequate. One way of counteracting the difference between situations is the use of transfer learning [150] enabling the use of the Chicago learning for the New York model, for example.

Choosing the Best Path using Pheromone Deposit

Mouhcine et al. [104] propose a MAS in which agents follow a pheromone-based strategy to indicate congestion on an axis. In their MAS called *"Distributed Vehicle Traffic Routing System"* (DVTRS), pheromones are dropped by stopped vehicles. Through communicating infrastructures, non stopped vehicles are aware of the congested roads and can therefore choose a more fluid road. The advantage of this system is that CAVs require little knowledge and communication to take their decisions. On the other hand, this MAS is not defined in a **hybrid context**: the CAVs will then lack information on non-connected vehicles. In this DVTRS, each vehicle is an agent with total control over its own decisions/actions and can be abstracted from the particularities of each scenario through the use of local knowledge only. The DVTRS evaluation done is **at the macroscopic level** and concerns the road network of a large city.

Pheromone deposit, inspired by the natural phenomenon of ants, is a way to share information without explicit communication. In the ITS context, this requires a means to depose artificial pheromone to enable a shared scenario representation between CAVs. A drawback of such a method is a lack of scalability when increasing the number of different information. In this work, only the information about congestion is concerned by pheromone deposit. But if more information needs to be shared, multiple types of pheromones have to be deposed.

Reduction of Shock Waves using Communications

A shock wave is often the result of intense actions such as emergency braking and it results in slowdowns and an increase in traffic density. Vaio et al. [151] propose a control strategy with MAS where CAVs coexist with non-autonomous vehicles, both being connected. CAV agents use the exchanged data to adapt their speed and to be informed as quickly as possible about the actions of others. Better anticipation before the shock wave prevents vehicles to act urgently: vehicles thus have a more fluid and more ecological driving behaviour. CAVs make local decisions using some data (velocity, direction, target) obtained every 20 ms. The experiment simulates vehicle dynamics in a **microscopic** simulation and shows that this strategy goes to **scale with a high number of vehicles**.

This exchange of information gives CAVs a way to anticipate others' behaviour using shared information. This work uses V2V communication and enables a local control without requiring a superior controller. It highlights that a CAV needs three different pieces of information (speed, position, and orientation) to take a

decision. The dynamic is intense and the flow of communication needs to support it. A drawback is that the scenario is known by all involved entities implying that CAVs know which information to send to others.

Reduction of Consumption by Anticipation

In the case of CAVs, communications can provide and process much more information than a human driver. The use of a predictive control model provides the vehicle anticipation capabilities. This type of model proposed by Taguchi et al. [73] combines lane change, acceleration, and braking data. In the context of perfect communications, CAVs accelerate optimally and predict the best time to change lane. But as data must be up to date, it implies a large volume of communications and in case of incorrect or missing communications, the system cannot work. A CAV does not have any knowledge of the scenario used for the experiment. The described scenario simulates six vehicles, on two lanes using a **microscopic simulator**, **scaling has yet to be proved**.

Anticipation is a way to avoid using under-efficient actions (strong break or acceleration) that harm the overall road traffic. Three pieces of information (speed, position and distance) are selected to improve a CAV's behaviour. Even with no knowledge about the scenario (this one being generic), useful information is known in advance. The data source and communication need to be ensured for proper functioning.

Selection of the Least Congested Intersections

The distribution of vehicles at intersections can be different and leads to traffic congestion. With the use of intelligent communications and infrastructure controlling intersections, CAV agents can decide to change the route as proposed by Lin et al. [82]. Their MAS approach allows a better distribution of CAV agents between intersections, leading to a **reduction of intersection congestion**. Each agent decides locally even if a server may ask it to change its route. Although communication is essential for the proper functioning of the system, **the volume of information exchanged is not high**. The CAV agent **knows the scenario** through the graph representing all the intersections given as input of the system. The presented experimentation **concerns the macro level and is scalable**.

This work presents a cooperative method based on V2I communication. The direction decision is given to the infrastructure instead of CAVs but they keep their autonomy if they do not want to follow the proposition. Communication requirements are reduced as infrastructures have increased and have specialised knowledge about their intersection. Moreover, vehicles have to share their objectives and such private information can be sensitive. The drawback is the non-inclusion of non-connected vehicles that can disturb the decided plan. However, this remains a good strategy on a large scale.

Cooperation of CAVs in Squads

Platooning is a squad-based vehicle formation with the front vehicle as the leader. The advantage of such a strategy is the fast and secure information sharing between the vehicles of the squad (Llatser et al. [84]). Thus, the vehicles follow the leader and adapt their speed to the leader. Vehicles can be closer than usual and do not risk collisions as they all act in a coordinated way; **the driving is smooth**. This type of training was experimented with real vehicles as part of the *"Grand Cooperative Driving Challenge 2016"* [34] in which two CAVs squads aimed to merge into a single squad. Once the merging action is initiated by leaders of each squad, each vehicle wishing to join the squad **binds by communication** to one of the present vehicles. Vehicles then insert themselves one behind the other. The *platooning* method **centralises decisions** (speed and travel) at the leader's level but each vehicle has autonomy for some actions, such as deciding to leave the squad when desired.

In traditional platooning approaches, each vehicle cooperates with the other platoon members through V2V as stated by Quadri et al. [121]. The main advantage of a pure V2V approach is that it does not require any network infrastructure. However, V2V comes with two important drawbacks: (i) as a consequence of the lack of a centralised scheduling policy, both interferences among vehicles/platoons and channel contention reduce the efficiency and the scalability of platooning; and, (ii), the communication range of on-board antennas and the shadowing of vehicles, whether or not they are part of the platoon, limit the length of platoons. In longer platoons, some messages should be relayed thus causing extra delay and increasing collisions risk.

Emergency Vehicle in Heavy Traffic

Driving emergency vehicles (EVs) in heavy or congested traffic is often problematic, as they are slowed down even if vehicles trapped in traffic try to clear the way for the EV. Agarwal et al. [6] propose a strategy for the EV that consists in choosing the fastest lane at one time and sticking to it while asking other Connected Vehicles (CV) to clear that lane. This strategy, constraining other CVs, is not efficient when the traffic density is too high. The authors proposed another strategy consisting in choosing the best lane according to the number of vehicles visible by the EV and the possible speed on the lane. It thus allows EVs to reduce their travel time. As only one message is exchanged between the EV and each vehicle, the volume of communication is low. The experiment of both strategies is carried out on a two kilometres road with the SUMO simulator [16] allowing **microscopic simulation of vehicles**.

This strategy scenario is similar to an obstacle detection that is shared by passing vehicles. Then, all vehicles share the same strategy to give way to an emergency vehicle. This is an example of CAV coordination policy enabled thanks to connected vehicles. However, this requires that all vehicles agree and can respect this strategy. Selfish or even opportunistic agents are not taken into account in these scenarios.

Anticipation of Lane Change in the Event of an Incident

When a vehicle has an incident that stops it in one lane, the reduction of available lanes might lead to traffic congestion. As a consequence, intense braking and lane changes at lower speeds may arise. To avoid this congestion, Ioannou et al. [107] propose to inform the CAVs in advance of the lane unavailability. The CAV agent, the incident initiator, or one CV receiving the message, can then disseminate the information. Although lane reduction still has an impact on traffic, congestion is reduced. Each CAV agent takes its own decision based on the received information and can communicate with other vehicles, Assistant Driving System (ADS) equipped or not. As only the presence of the incident is exchanged, the communication volume is low. A microscopic experiment is presented and this **system succeeds in scaling up**.

This study highlights the contribution of communications, and the use of local control for a CAV to make better decisions. Many strategies have been developed, based on communication and bring improvement on traffic flow. Communication between CAVs only and/or with infrastructure is primordial to enabling cooperative strategies that help to reduce congestion. In many works, communication takes place only in a local manner making the decision made more suitable for the local problem.

2.7.3 Panorama of Fleet of CAVs for Intersection Management

An intersection may be seen as a dangerous area because several vehicles have to share the lane that they sometimes cut off from each other. Until now, the road code and traffic signs were sufficient to guarantee a minimum safety of users to manage intersections when all the rules are respected. In this section, the studied CAVs systems addressing this issue are using two different approaches. First, the centralised approach with an agent *Infrastructure Management Agent* (IMA) in charge of the intersection and which concentrates decisions.

Second, a decentralised approach in which CAV agents cooperate to find a consensus, with an IMA possibly helping this cooperation.

Coordination of Vehicles by Reservation in an Intersection

In the centralised approach, an IMA communicates with vehicles wishing to cross the intersection by indicating how to proceed : it uses the notion of reservation [33, 69, 118]. Jin et al. [69] propose a Multi-Agent Intersection Management System (MAIMS) using reservations. It means that the IMA reserves a slot that allows the CAV to cross the intersection without conflicting with other vehicles. The CAV agent approaching the intersection has to request a reservation by sending its objective. The particularity of the MAIMS is that it uses a vehicle scheduling approach instead of a FIFO one. All the near CAV agents report their speed, position and destination to the IMA that then organises their crossing order. Authors have noticed that with a large number of CAVs, the volume of communications becomes problematic. The experiments carried out on SUMO with MAIMS show a clear **improvement of the vehicle flow** compared to the FIFO-based approach. However, this system shows no improvement in fuel consumption: the authors suggest that it comes from vehicle trajectories. The presented experimentation uses a large number of vehicles: the **scaling for this scenario is validated**.

Optimisation of Trajectories in an Intersection

Kamal et al. [72] propose a Vehicle-Intersection Coordination Scheme (VICS) to obtain more fluid traffic at an intersection and to reduce fuel consumption. To reach these objectives, the trajectory of each vehicle agent is optimised using an IMA located in the intersection, which schedules the CAVs. The authors experiment with VICS on Matlab, which is **far from real vehicle conditions**, and they showed that VICS eliminates almost all the stops at the intersection: this **significantly improves traffic flow** and **fuel consumption** for each vehicle. An experiment with a large number of CAVs shows **the transition to system-wide** and thus being scalable. Note that VICS operates in the context of a **traffic composed solely of CAVs** that are cooperative with the IMA, so it does not take into account a **hybrid environment** and the **volume of communications may become too large**.

A centralised approach leads to significant data exchanges with one entity and possible congestion of communication flows. CAVs, equipped with V2X perception and communication devices, can also negotiate to facilitate the resolution of blocking situations. Decentralised approaches aim at giving control to the vehicles instead of a superior entity. Compared to a centralised approach, the overall amount of communication increases but with fewer chances for a bottleneck to occur.

Negotiations between CAVs at an Intersection

Gaciarz et al. [44] propose a negotiation mechanism between CAV agents wishing to cross an intersection. In this mechanism, the intersection is divided into cells and CAV agents communicate to obtain a coherent configuration. The IMA of this system has greater knowledge than CAV agents and can propose configurations that improve the overall fluidity without impacting local agent satisfaction. Each agent is free to accept or refuse the IMA proposition according to its preferences and objectives. Negotiation includes all the agents present in the intersection at the current time and it does not generate a high volume of communications; however, the presence of non-cooperative vehicles is not considered. Experimentation with this mechanism shows that the length of queues in the intersection is reduced. The simulation using only *threads* with no real micro-behaviours does not allow **to have a simulation close to reality** but the number of vehicles is sufficient to **validate the scalability**.

CAVs keep local control on their decision and are not completely submissive to another entity. Negotiation may be difficult to rapidly achieve and may cause an issue when no agreement is found. Shared data are endogenous ones only, meaning that any information about the intersection may be missing. As endogenous data are assumed reliable, noise is not considered.

2.7.4 Anticipation of the Traffic Light State

As a traffic light at an intersection controls the flow of vehicles in the axes, anticipating the state of an intersection managed by a traffic light helps to avoid unnecessary stops and intensive braking. Communicating infrastructures allow a CV to request information on the state of the intersection to behave optimally. Almannaa et al. [10] have experimented with a **real vehicle** in a connected intersection. When the vehicle is within the communication range of the infrastructure, the traffic light communicates the remaining duration of its condition to the vehicle, which can locally adjust its speed with the objective of not stopping at the intersection. This experiment shows that with this information, a CAV can adapt its speed to the intersection state, thus reducing its fuel consumption. Communications necessary to the proper functioning of the system consist of **a single message** indicating the intersection status.

These studies underline that the CAVs can negotiate together the appropriate time to cross an intersection with a low volume of communications. Information exchanges between CAVs and infrastructure improve the smoothness of the crossing.

2.7.5 Panorama of Fleet of CAVs for the Management of Lane Merging

The third situation that a CAV must be able to handle is the lanes merging. This type of situation is difficult to manage because a vehicle that wants to join a lane has to adapt its speed to the other vehicles in that lane but also to find the best place to insert with a minimum of discomfort for other vehicles. Human drivers have trouble handling this situation because of mutual-understanding problems. This often comes from the driver's belief of what the other is intending to do. CAVs can communicate directly with each other to clarify their intentions.

Coordination of CAVs at Merging Point

Rios-Torres et al. [123] propose that the merging zone of the two lanes is managed by an agent present in a connected infrastructure. This controller assigns an identifier to each vehicle to organise them in FIFO for this area. The controller ordering the insertion allows the management of CVs. This system seeks to optimise fuel consumption by calculating the most appropriate time and speed to insert smoothly and safely at the merging point. Experiments with this system can include a high number of vehicles without being too computationally expensive. However, the authors highlight a **limit when the vehicle speed is too high**.

The use of a controller is less problematic for private information in this scenario type because all the vehicles have a common goal, to move in the same direction. However, this proposed strategy creates a plan which does not include errors or abnormal behaviour. One malicious vehicle can disturb all the proposed plans.

Cooperation Upstream of the Merging Point

Wang et al. [158] propose a protocol for cooperation between vehicles before reaching the merging point of the lanes. Each CAV agent sends its data (acceleration, speed, and position) to an infrastructure agent that continuously sequences the vehicles and indicates to each of them the vehicle in front. Then the CAV agent receives data from the vehicle in front and has to adapt its speed to reach the merging point at the optimal time

(known in advance), even if both are on two different lanes. An agent-based simulation made of **six vehicles** in *Unity* [57] compares the use of this system to human conductors; it results in a **reduction of travel time and energy savings** for CAV agents. However, a **large volume of data** is required for this system.

Although these two previous approaches present very interesting results, it requires the presence of a controller and therefore an infrastructure to host it. The use of an IMA enables the perfect coordination of vehicles in the intersection. However, they are only possible in a fully autonomous environment and are subject to unpredictable events that may disrupt the plan. Three decentralised approaches to avoid infrastructures are now presented.

Decentralised Cooperation before the Merging Point

Mosebach et al. [103] propose a decentralised lane merging control algorithm in which CAVs communicate to determine if a vehicle is in the area preceding the lane merging. If a CAV has detected at least one vehicle, it slows down and calculates a trajectory to cross the merging point without collision. When the CAV considers that its trajectory is correct, it follows it by re-accelerating. The required communications include the other vehicles as well as their speed. In this simulation, the CAVs **know the road typology** in advance and cross the merging zone in FIFO. A two-step process is followed in this strategy. First, communication is used to extend the perception of other vehicles thanks to the communication range. Any connected vehicle in communication range is indeed perceived when it sends back its position and speed. Second, using the position and speed, an accurate strategy is computed. This cannot work with non-connected vehicles because they do not answer back (with all consequences that imply). Two possibilities must be considered: 1) if there are other connected vehicles in the objective lane, this one could send the wanted information, or 2) having a backup strategy for the merging point not relying on communication. The simulation of two lanes merging with **40 vehicles** shows that CAVs are inserted **smoothly** while maintaining a safe distance.

Decentralised Merging in Mixed Traffic

Sobieraj [141] proposes a cooperation protocol between CAVs to enable CAV to merge in mixed traffic. In an area known in advance by the CAV agents and close to the merging point, the CAVs of the two (merging) lanes communicate to form pairs. Then, the CAV agents of each couple coordinate their acceleration to position themselves at a safe distance one behind the other. **The resulting communication volume is low** and acceleration calculations are based on models that allow smooth insertion. The experimentation consists in merging two lanes crossed by a dense vehicles flow. The presence of CAV agents **improves traffic flow**, and waiting times have almost disappeared.

This approach can be used in mixed traffic despite lower performance is interesting but is of course still limited to connected vehicles for the cooperation part. The presence of non-connected vehicles is a problem for two reasons: 1) they cannot be a part of the cooperative strategy and 2) they may disrupt the actual cooperative strategy that CAVs intend to make.

Cooperative Change Path with Confidence Index

Monteil et al. [101] have designed a cooperative mechanism between CAVs agents, which enables to consider unreliable (or absence of) communications. This method is based on the use of a confidence index on the information given by each CAV agent. Thus a CAV agent calculates the confidence index for each vehicle communicating with it and thus may evaluate the quality of the communications. In the case of missing or unreliable communications, the CAV only uses its perceptions to make decisions: **the presence of unconnected vehicles is considered**. The **local control** and the ability to use only its perceptions **validate**

the robustness of this method. Adding to this mechanism a connected infrastructure [54] provides access to additional information to the CAV agents with an **increase in the volume of communication**. The simulation of a flow with **a large number of vehicles** shows that the presence of CAVs and infrastructure **smooths** insertion in the merging zone.

These two works present two interesting characteristics. A CAV can identify unreliable communication and change its strategy to use only its. The second is the use of another entity's communication to have access to extended information about the situation.

2.7.6 Synthesis

Table 2.2 summarises and synthesises the results of the previous study (with ratings ranging from - - to ++; - - means *not at all* and ++ *totally*, while NA means *not addressed*). This table helps to understand the scope of all the works by highlighting their characteristics, their limits, and their strengths. Even if not all existing works are represented, due to their large number, it is difficult to design a fully functioning fleet of CAVs.

It can be noticed that the majority of systems only work in a fully connected and cooperative environment. The presence of unconnected, so eventually uncooperative vehicles, prevents CAVs to determine the behaviour of these kinds of vehicles, and consequently, they cannot optimally implement cooperation. Nevertheless, as CAVs and non-ADS-equipped Vehicles will very probably have to coexist before the traffic becomes fully autonomous, this is a strong limit for these systems to bring improvement. It is easy to see how a non-cooperative entity can disrupt all systems. Worse, a malevolent one has even more potential to cause damage knowing how other CAVs behave. One thing that can be added is that in all studied solutions all CAVs **understand each other** and **communications are almost always reliable**. In conclusion, to enable the majority of strategies, these two assumptions of understanding and reliability must be matched.

Only the system of Wang et al. [158] has an explicit influence on the **travel time, emitted pollution, and fluidity of the traffic flow**. However, it only concerns the insertion in a lane merging. Systems aiming to find the best route mainly influence **travel time** even if it is not clearly stated. There are many possibilities to improve one or several road traffic characteristics (traffic flow, pollution, etc). Each proposed strategy can be included in a fleet of CAVs because bringing an improvement to the overall road traffic state. With recent and future advancements in a fleet of CAVs, the number of strategies is likely to increase. This may lead to embedding in each CAV all designed strategies, contextually dependant.

Managing an intersection requires a substantial **volume of communications** when there are a large number of vehicles. An IMA should be seen as a tool that facilitates cooperation in a fleet of CAV as it could not equip every intersection and it can have failure. The studied systems need either IMA or CAVs agents to have some scenario knowledge. There are two possibilities to acquire this knowledge: 1) having a source providing the intersection structure or 2) gathering enough information to build an accurate representation of it. Requiring knowledge about intersections has another consequence: when the intersection structure is modified, the specification must be modified (the knowledge of the IMA or the CAV). The second possibility is more suitable to adapt to dynamic changes but requires an increased exchange of information.

Only the systems whose objective is to find the best path are **robust** because they do not require making a real-time decision with only 1 second to process. This is due to the complexity of the real vehicles and all the factors that need to be considered when decisions are made within a very short time frame. As a critical system evolving in the real world, it is impossible to know in advance every event. Robustness can be ensured with a critical decision. In robotics, this is often achieved thanks to an emergency stop when the system reaches a state where no other option is available. This is problematic for two reasons in the CAV context: 1) an emergency stop can cause more issues and 2) this brings discomfort. The latest one is less critical but it is as important to address it. One of the challenges is to improve the quality of life to be accepted by road users. If a vehicle has to stop too many times in a hurry or to make passengers feel not safe then people will not

accept to be driven by this vehicle.

In all three situations, the fact that a CAV agent shares its objectives, speed, and position allows other vehicles to have better anticipation and understanding of this vehicle. This allows cooperative strategy to take place. However, CAVs may conflict with one another due to their differents objectives. But it is also possible that conflicts come from an misunderstanding or an error in communications. Indeed, all information shared is dependant on the referential used. Studied works assume that the referential frame used by all CAVs is common, but it cannot be guaranteed for now. This can be an obstacle to the proper functioning of the fleet.

A CAV that only needs local information and a small amount of knowledge can adapt to unexpected situations. As shown in Table 2.2, only systems with a positive assessment for these two criteria also have a positive assessment for robustness. Moreover, using local behaviour helps to prevent the sharing of private information that can be harmful. The natural control of the distribution of CAVs leads to the use of local strategy, to avoid the bottleneck of control, private information exchanges and latency.

System	Travel Time	Consumption and Pollution	Smooth Driving	Locality	Communication Volume	Robustness	Scenario Knowledge	Mixed Traffic	Scalability	Simulation Model
Traffic Flow : Choose the Best Path[1] or Smooth Driving[2]										
Barthelemy et al.[1]	+	NA	NA	++	+	+	+	+ -	+	Macro
Mouhcine et al.[1]	+	NA	NA	+	+	-	+	-	+	Macro
Di Vaio et al.[2]	NA	+	+	++	-	+	+	+	+ -	Micro
Taguchi et al.[2]	NA	+	+	++	-	-	+	-	-	Micro
Lin et al.[1]	+	NA	NA	+	+	-	-	-	+	Micro
Englund et al.[2]	NA	NA	+	+ -	+ -	+ -	+	-	-	Real
Agarwal et al.[2]	+	NA	+	- -	+	+ -	-	-	-	Micro
Ioannou et al.[2]	+	+	NA	++	+	+	+	-	+	Micro
Intersection										
Jin et al.	NA	-	+	- -	+ -	-	+	-	+	Micro
Kamal et al.	NA	+	+	- -	+ -	-	+	-	+	Micro
Gaciarz et al.	+	NA	+	+	+ -	+	+ -	-	+	Macro
Almannaa et al.	NA	+	+	++	+ -	-	-	- -	-	Real
Lane merging										
Rios-Torres et al.	NA	+	+	+ -	+	-	-	+ -	-	Micro
Wang et al.	+	+	+	+ -	-	-	-	+ -	-	Micro
Mosebach et al.	NA	NA	+	++	++	+	-	-	+ -	Micro
Sobieraj et al.	NA	NA	+	++	+	-	+	++	+	Micro
Gueriau et al.	+	NA	+	++	+ -	+	-	++	+	Micro

Table 2.2: Synthesis of studied fleet of CAVs Systems

2.7.7 A Guideline to Satisfy the Requirements for a Fleet of CAVs

From the requirements and the analysis of the different systems, a guideline to design a fleet of CAVs is proposed, which consists of high-level design principles concerning decentralisation, communication, robustness, knowledge of the scenario, mixed traffic, and scalability. All the CAVs in the studied systems have the same design and consequently the same behaviour in a given situation. However, in real life, they have different designs leading to several behaviours in the same situation. Thus, the results given in these studies have to be taken with hindsight.

Centralisation can be seen as ”good” or ”bad” depending on the objective. Because centralised systems use a lot of information considered as safe, regardless of uncooperative behaviours or incidents, **a local control of CAV** using partial perceptions and being able to adapt to unexpected situations is mandatory. To avoid bottlenecks of communications, the use of a centralised system for communications should be avoided. However, each CAV should not share all its data with all other CAVs because the volume of communications would explode. It is required for a CAV **to learn which information is useful** to share for a situation to limit their amount and to facilitate the decision process.

Even faced with an unknown situation, a robust CAV must be able to take the correct decision. Moreover, it should be able to learn which behaviour is better for the next time a similar situation will be encountered. Because the amount of information available is continuously evolving and it seems unlikely that the same data would be expressed in the same way by two different vehicles, a CAV should **learn continuously the usefulness of information without semantic**. At first, the CAV would drive safely, like a new driver, and it will gradually acquire experience while driving, improving its behaviour. A CAV that aims at managing every road situation needs to be able to exchange its data and negotiate with other vehicles. Moreover, it needs to be able to cooperate with other connected entities to refine its knowledge of the context and thus, it will lead to having a better behaviour [14, 151].

It seems that situation awareness can be a barrier because it is impossible to model all the possible situations beforehand. A CAV should be able to understand the current situation so that it can adapt its behaviour to it. Coordination between vehicles will make it possible to solve many traffic situations. So, **cooperation and negotiation must be implemented in each CAV** [44, 84, 158]. The presence of non-CAV could be addressed by considering them as non-cooperative entities with unpredictable behaviour. The CAV must be able to **identify them and learn how to interact with them**.

In the validation step, the evaluation of a fleet of CAVs must be done **simultaneously at the micro and macro levels**. The first one is necessary to observe how a fleet of CAVs manages a precise situation while the second one is used to observe the result at a larger scale.

2.7.8 Analysis

The previous study of existing works about a fleet of CAVs gives an overview of how it can have a positive impact on road traffic. The number of possible different road situations can be considered as non-finite. Indeed, most of the works focus on an abstract representation of a road situation. Often many elements are missing and those elements are what make a situation more complicated than usual. Usually, when studying a specific aspect of road traffic, this one is simplified and does not include every possibility. Such a solution is indeed convenient but looking only at one aspect at a time may result in the omission of phenomenon coming from the multiple interactions happening at the same time.

This is quite problematic as studying only perfect cases is not a way to ensure the reliability of a system. When looking at all elements, the number of information that a CAV can perceive increases strongly. Some car manufacturers have provided videos of their detection algorithms. Using the video flux of **one** sensor, the algorithm can detect many objects with one image. And from the objects, much information is extracted like speed or direction. Now in a complex situation with many vehicles, many communication may take place. The question is how a CAV decides which information may help some other CAV? In most of the presented works, this selection of information is already known, and it is the only one sent. An answer could be ”send everything” but at what point does the total amount become an issue? Moreover, receiving the same information 10 times may be a problem.

There is another point missing from common works on a fleet of CAV. Often an object position is shared using communication. The common referential used is GPS positioning. Currently, GPS has a precision of two meters and is expected to reach a precision of 30 centimetres in 2027 using 24 satellites [102]. However, the only position is not enough for certain information. Moreover, receiving its position is not the same as estimating the GPS position of another object. Another issue with GPS is the underground situation like tunnels, areas without or with little satellites, and potential failures. For all these reasons, only relying on GPS is not a robust solution. It needs to be covered with additional features.

Once GPS is excluded, it is difficult for a vehicle to share a common referential. Moreover, if no standard is enforced by an authority the same information can be expressed in different units. Even without a shared referential, CAVs must be able to communicate efficiently. A vector with position, direction, and speed is

subject to a referential. When perceiving it, a CAV uses its own. Sending as it is may cause misunderstanding or errors. This must be avoided at any cost. A referential frame may also be subject to an automobile constructor particularity. For example, a difference in how the speed is measured (km/h or m/s).

Among the 17 systems presented, 12 are based on MAS in which CAV agents use a cooperative process when confronted with one of the three situations. MAS is an approach based on the local behaviour of entities called agents. A sub-part of this approach uses cooperation as a means to obtain a functional system. Cooperation is enabled thanks to the interaction between agents. In the ITS context, this requires communication to be reliable and understood. Consequently, a challenge is that a **communication must be understood** by any members of the fleet. Every work brings improvements to at least one road traffic aspect; in consequence, every one of them should be included in a CAV's decision process. Even if communication volume remains low for many works, the total amount of every communication, that must be up-to-date, increases fast with the number of strategies to give to a fleet of CAVs and the number of CAVs. Indeed, each CAV may want to exchange the same information, thus leading to an exponential increase in the number of messages. In consequence, the **exchange of information must be optimised**.

2.8 Conclusion on fleet of CAVs inside ITS

Intelligent Transport System and particularly fleet of Connected and Autonomous Vehicles is a complex domain involving many research fields from the material construction through robotics controller to artificial intelligence. Advancements in this domain are expected to improve people's quality of life ecologically and comfortably. Urban areas are the most likely to be affected by these changes. Thanks to recent advancements both in Artificial Intelligence and in mobile Network Connected and Autonomous vehicles are not only a dream for the future. Even if in a theoretical way nevertheless, this field has been heavily studied in recent years. Without mentioning progress that automobile constructors have shown to reach level 3 of automation.

Road traffic is a complex environment. This complexity varies according to the encountered road infrastructure, time, weather and other more specifics rare events. Then, designing upstream all possible events is not possible. This is why a learning or adaptive strategy is often preferred. A CAV is a critical system that may cause casualities in case of bad behaviour. Moreover, it needs to be accepted by other road users.

Most of the works that can be found in the scientific literature highlight benefits that using CAV bring. With success, these works propose a strategy/method/system to improve a specific part of the road traffic. Two main paths are used: 1) using a controller, gathering every CAVs' information, and using this massive information amount to decide the bests actions. Efficient in a way, such a solution is often limited for large scale, not fully CAV environments and privacy concerns. Moreover, no system is perfect so a controller's failure may happen to put all the systems at a stop. Or 2) leaving the control to CAV with limited perception. The principle of a fleet of CAVs is to see all vehicles inside as cooperative. Through V2V communication they achieve a greater understanding of the situation. From this, their performed action is better while keeping local control.

Communication is a great way to share information. However, as a complex system, this number of possible exchanged information can reach an amount large enough to be problematic (even more, if the same information is communicated by all involved vehicles). Network progress show more and more efficiency to support fast, reliable, and large communication. But it can be problematic for the CAV. Indeed, a decision must be taken within a short time and failure is not acceptable. With too much information communicated, a CAV may not have the time to process all incoming messages before it has to make a decision.

Another problem comes from the sharing of information. Without a common referential frame, the exchange of a vector position, direction and speed are subject to errors. The difference in sensors or driving processes may modify information according to the referential.

These two last points need to be addressed generically. A CAV is not yet finalised and is subject to modification. No systems remain the same until the end of time. The next chapter presents a state-of-the-art of standard techniques that can be used to address challenges of **understanding an information** or **communication optimisation**. These are not new topics but they are already a part of the **Data Estimation** and respectively **Constraint Optimisation** fields. Both domains are precised and studied in the next chapter to have a proper understanding of what are the challenges.

Chapter 3

Methods and techniques for cross-understanding and optimisation of communication

---

Objectives of this chapter:

- From the scientific literature are presented existing techniques and needs to address both cross-understanding and communication optimisation. It is shown that adapting an information in another referential frame is seen as a data estimation problem and that communication optimisation can be approached as a Distributed Constraint Optimisation Problem.

- Each technique highlights which characteristics enable to address the challenges of dynamic, openness, privacy, and scalability.

This chapter aims at providing an accurate study of the communication in a fleet of CAVs according to the objectives of **cross-understanding** and **communication optimisation**. The challenges studied in this thesis can be summarised by these two questions: 1) how to understand each communication and 2) how to ensure that every communication can be processed.

Many strategies to improve road situations are based on communication. This communication is supposed to be reliable and useful. To be considered useful, communication must increase the receiver's knowledge. From this thesis perspective, it is when 1) the communication is understandable and 2) it brings an increased knowledge for the receiving CAVs. However, many causes prevent the robustness of these assumptions. This chapter is constructed by first giving a precision on communication challenges for social entities: **cross-understanding** and **communication optimisation**. A study of the techniques to address how such challenges can be solved is then proposed with a focus on **dynamicity**, **openness**, **scalability**, and **privacy**.

The first analysed technique domain is **Data Estimation** that can be used to solve the problem of communication understanding. The second studied technique domain is the well-known **Distributed Constraint Optimisation** framework. This latter is studied in the context of optimising communications, i.e. selecting the most useful information without creating a cacophony (confusion of information). Standards algorithms are also studied and their adequacy to the problem is analysed.

3.1 Communication in a Fleet of CAVs

The emergence of communication has been long studied in History. The Pharaoh Psammetique I (663-610 BC) has entrusted two children to a shepherd and forbade him to speak to them. The objective was to wait until the children spoke without anyone to learn them how to communicate. Children were able to speak. The same experiment, closer to us in time has been conducted by the emperor Frederic II (1194-1250). Tragically, the experiment resulted in the death of the children who never spoke. It seems that without social interactions, the children did not spontaneously develop a language (verbal or not) [74]. Nowadays, an observation (such experiments on children have fortunately been forbidden) on a school of deaf children [75] has shown the emergence of a particular type of communication. Although no one taught them, sign language has emerged between children. Those three studies show that communication codes emerge from the interactions between social entities and develop through time. So, the constructed language is the result of the multiple interactions within a social group.

In the context of this thesis, the communication is studied to observe how several heterogeneous CAVs exchange information with one another. The objectives are to see how this communication impacts a fleet of CAVs but also, how a fleet of CAVs can be an effective social organisation. The particularity here is that communication has to be useful for a CAV to share it. But a CAV cannot know upstream if the information it has is useful to others. This is why a CAV needs to select what piece of information to exchange and which one to keep because useless messages are seen as noise that disrupts other entities' behaviour and must be avoided as much as possible.

This section first presents the notion of a referential frame for a CAV and the difficulties it may induce. Then, the techniques characteristics that are looked for are extracted from the fleet of CAVs context. These characteristics are then studied in the rest of the chapter.

The Referential Frame of a CAV

Each CAV decision is taken from or by taking into account its environment observations. The observation of an object is dependent on the observer. A good illustration is an image that went viral on social networks in 2015 about the colour of a dress [91]. This dress was seen as blue by a part of the population and white by the other. If this is observed by an autonomous entity that stores this information in its knowledge base, the result would be $Dress + Blue \rightarrow BlueDress$. Then, this entity exchanges this information to another as "there is a blue dress". The other has seen the dress, but has stored it as "WhiteDress". The conclusion from the example is two possibilities: two different dresses or only one but seen differently. Trusting the communication reliability the receiving entity adds to its knowledge base the blue dress as a different object. In consequence, the knowledge base became erroneous. Two CAVs see the same object from their respective point of view that may be different. This example illustrates the concept of "**referential frame**" studied and discussed in this thesis. The objective is to enable CAVs to understand what the other communicates in any case, this is called **cross-understanding**. The definition of referential frame used in this thesis is the following [78]:

Definition 1 (**Referential Frame**). *The referential frame of an entity is the plan used by this entity to position elements of its environment with itself at the centre. An object representation is modified according to three factors:*

- *The relative position of the object for the entity;*
- *The quality of perception related to the entity's sensors and also obstacles encountered by the entity;*

- *The rate of the perception affecting the collect time. A slow perception rate may lead to missing modification in the environment. In the case of rapidly changing value, the information may no longer be accurate.*

The first objective of this thesis is to enable a CAV to understand the way other CAVs see their environment to enable a **cross-understanding** between CAVs. More specifically this objective is to propose a solution for transforming data from a referential frame of a CAV into a referential frame of another CAV, to make them usable by both of them.

The number of information that can be perceived in road traffic is huge. Each CAV communicating each perceived object to every other road user is similar as if each person in a meeting talking out loud at the same time. In consequence, no information is retained and communication is useless and not usable. Moreover, communication has a cost on energy, Sun et al. [144] investigate communication minimisation for cooperative unmanned aerial vehicles and they looked precisely at the energy consumption required for communication. In the fleet of CAVs context, the necessity of optimising communication is justified by the computing capacity of the vehicle. Because some information is redundant or useless, a quality improvement is to optimise the shared messages. So, the second objective of this thesis is to **optimise the communication** between CAVs to only communicate the most useful information without exceeding the capabilities of CAVs.

Analysis of related works on communication optimisation is conducted according to four properties: dynamics, scalability, openness, and privacy. These four properties have been chosen from the ITS context characteristics and are explained hereafter:

- **Dynamics**: the road traffic is highly dynamic. Looking from a fleet of CAVs perspective, there are many evolutions during driving due to the diversity of entities (non-CAV vehicle, pedestrian, animals etc). Moreover, the real world is continuously evolving with weather changes or other perturbations. To be efficient a technique must be able to address dynamic evolution.

- **Scalability**: According to the ITS context including a huge number of vehicles, a technique must take into account that the number of involved entities may be large. Moreover, the information processed volume and used in a CAV decision is increasing with the number and quality of sensors.

- **Openness**: A fleet is made of several entities (CAVs), each one having its objective (the final destination). The fleet composition is not fixed and several CAVs may enter or leave the fleet over time.

- **Privacy**: As an autonomous entity, possibly including users' preferences and data, a CAV must limit the amount of personal information diffusion. The amount of private information shared must be limited.

3.2 Cross Understanding

This section addresses the notion of cross-understanding. The Cambridge dictionary defines the two words composing it as follows:

- **Understanding:** "knowledge about a subject, situation, etc. or about how something works".

- **Cross:** "affecting two people equally, or felt equally by two people". This definition implies the need for the information to have the same impact on the knowledge for each concerned part.

From the previous definitions, two entities that **cross-understands** each other, extract the exact same knowledge about an information they both have. Achieving cross-understanding between two social entities is a complex task. To be able to understand someone else clearly, not only the knowledge is required but also the

way this knowledge is expressed needs to be known. In the context of a fleet of CAVs, this is illustrated using the reference frame of each CAV. Two CAVs have to be able to understand each other but their difference of position modify their perception of the world. An object position is sensed according to the CAV's position, so the same object position is seen differently. Suppose two CAVs, CAV_A and CAV_B, coming in opposite directions. CAV_A detects a pedestrian crossing the road, 10 metres ahead whereas CAV_B detects the same pedestrian crossing the road, 20 metres ahead. The knowledge of CAV_A is communicated to CAV_B. If CAV_B processed the information sent by CAV_A as it is, the position and the distance of the object will not match the position and distance of this object from the point of view of CAV_B. In addition to the position, several parameters of the information must be taken into account, such as units (meters, miles, kilometres etc), sensors characteristics (precision, specificities, etc). Thus, before using received information, the CAV (CAV_B in our previous example) needs to transform the received information for its own referential so that this information is useful. The difficulty for a CAV will be to find/determine if the received information matches one of its perceived ones and to which one? To summarise, to achieve cross-understanding, two aspects need to be taken into account: 1) to select correct information from a large set of information and 2) modify the referential frame from other information to be usable. To overcome the problem, several/numerous works tackle the problem of **Data imputation** which is discussed hereafter.

3.2.1 Data Imputation

The objective of Intelligent Transport Systems (ITS) is to improve the traffic to make it smoother, safer, etc. And consequently more comfortable, faster and cleaner. Vehicles are equipped with an increasing number of sensors that make the vehicles intelligent. The sensors data from a CAV is crucial for the CAV to make the most appropriate decision for the situation at hand. Data acquisition is key to making a decision and lack of data prevents an optimal decision, which leads to less efficient behaviours and can even be dangerous (as shown by some real-life examples where an autonomy level 3 vehicle was not able to detect the presence of a pedestrian [46]). ITS aims to prevent this by allowing intelligent road entities (vehicles or infrastructure) to share their perceptions through dedicated networks and protocols. As a CAV can be unable to perceive the information required in its surroundings, thus its knowledge would be missing an information. As a system with a large number of entities, a missing information for one entity is likely to be perceived by a subset of other entities. In the absence of a common referential frame, the information description is ensured to be correct only for the entity that perceives this information, the other entities each having a different referential frame. To overcome this issue, two strategies can be used: 1) sharing the same referential frame or 2) estimating the transformation from the previous observation. Sharing the referential frame to merge two environment representations has the major drawback of being heavy in terms of communication [5].

The estimation of missing values in a dataset has been studied since the seventies as a technique to address data incompleteness [92]. Missing data from a dataset may have several sources like faulty sensors or environment chaotic behaviour (for example mud on the sensor). It can also be beneficial to **increase** the number of values of a dataset, that will be used by data-hungry algorithms (as deep learning) or second to apply a patch on errors. Estimating missing data is a challenging task in domains that need real-time and on-demand estimation.

Missing data problems are classified into three categories [126] that influence the choice of method used to address them:

- Missing Completely At Random (MCAR) means that the causes of the missing data are unrelated to the data. An example of a cause is a sensor running out of battery. MCAR is often considered unrealistic for the data in the real world [152].

- Missing At Random (MAR), a larger category than MCAR, assumes that the cause of missing data (i.e. an obstacle in sight) is known. This is the most common category assumed.
- Not Missing At Random (NMAR) includes all the cases that do not belong to MCAR and MAR. This category, the most complex one, assumes that there is an unknown (or a difficult to process) cause of missing data. An example would be wear and tear on a sensor that produces more missing data during use. Methods addressing this category focus on understanding the cause and/or the sensibility.

Two categories of methods address data estimation: **mathematical methods** (Regression) and **Artificial Intelligence** methods. Both of them are detailed and analysed in the two next sections. The analysis is constructed from the following challenges. Estimated information is used in the decision process of a vehicle, the value must be close to reality and computed within a short time. The **scaling** characteristic is also a main focus as the system includes many different entities with a high number of information. So it has to be fast in the computation of many data and not be overloaded with an increasing amount of information. The ITS is highly dynamic with many entities moving in real-time, so the method used has to be able to adapt to **dynamic** changes and be **open**.

3.2.2 Mathematical Data Estimation Methods

The Mathematical Data Estimation methods use a mathematical function to predict a missing value, which is a solution fast and easy to implement [152]. Two main techniques have been designed, one based on the mean estimation and the other on a regression estimation.

Mean Estimation

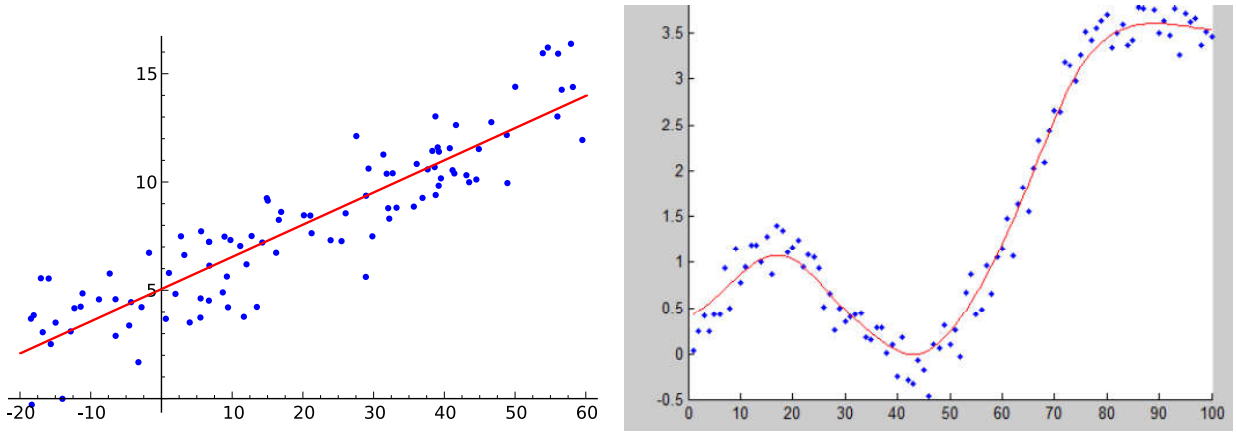
The mean estimation technique is considered the easiest way to rapidly provide a missing value in a dataset, but not the most accurate. Missing values are calculated using the mean of existing values. The mean calculation is altered by uncorrelated variables thus leading to erroneous estimation. This technique is not the most reliable and other techniques should be preferred [152].

Regression

Regression analysis belongs to the mathematical domain used to model the relationship between a scalar (dependent) response and one or more explanatory (independent) variables. With only one independent variable, the regression is called *Simple Linear Regression*. A simple regression equation has on the right-hand side an intercept and an explanatory variable with a slope coefficient. Multiple regression has several variables on the right-hand side, each with its slope coefficient [42]. Models can also include noisy variables by adding a coefficient to the equation. Formally a coefficient μ is added to the right-hand side (see equation 3.1). x_0 is the value that has to be estimated, x_1 is the explanatory variable, *a* is called the *intercept* and *b* is the slope coefficient. It is possible to have several independent variables, in this case, the regression is called *Multiple Linear Regression*. The strength of such techniques lies in their accuracy. This is achieved thanks to a proper calibration of the model. However, because a calibration step is required, those methods have difficulties taking into account openness. The integration of a new device in the system induces the need to re-calibrate the model.

$$x_0 = x_1 \times a + b + \mu \tag{3.1}$$

Two steps are required to build a regression estimation. The first step of a regression estimation consists in building the model from data observed before they are missing. Once the model is fitted, it can be used for



(a) Example of simple linear regression, which has one independent variable

(b) Example of polynomial regression, which has one independent variable

Figure 3.1: Examples of regressions techniques.

the second step. From the value of an available independent variable (a selection of best variables according to the model), the model estimates the value of the missing one (used as a dependent variable). The linear regression model has for objective to define a linear function as it minimises the error on the overall dataset for the considered variable. Its graphical representation is a straight line in figure 3.1a. However, the variation may be too wide to be addressed using a linear function. In that case, a polynomial regression can be used, which is graphically represented by a curve like in figure 3.1b. Similarly, as with linear regression, the aim is to minimise the error with all values in the dataset. Polynomial regression is more complicated than a linear one but allows the use of curve functions. The use of curve functions allows a better match with the points cloud, and the greater the degree of the polynomial, the more accurate the curve. The estimate, although more complicated to find, will be more accurate.

Regression is simple and efficient to estimate missing data values but is still an active research field. Hereafter are presented two recent examples using regression models to estimate missing data in ITS with success.

Spatial Distribution of Air pollutants in Urban Environments

Hasenfratz et al. [58] propose a solution taking advantage of the vehicles' mobility to create mobile sensors to collect several data (pollution, traffic,...) to determine a model of the spatial distribution of air pollutants. They use linear regression to find correlated variables and build the regression model. From this, they manage to predict the pollution concentration level at precise locations. Authors use a special regression model, Generalised Additive Model which is particularly efficient for environmental variables.

Fusing Incomplete Multisensor Heterogeneous Data to Estimate Urban Traffic

Shan et al. [137] present a regression method based on a Multiple Linear Regression (MLR) model to fuse multiple values/data collected by sensors to replace missing data. Their objective is to improve the estimation of traffic state using GPS embedded on taxis on three road segments. This method extracts the spatio-temporal correlations of traffic state from the GPS data of the vehicles. This extraction uses a Least Squares Method when the dataset is complete, a Heuristic Estimation Method otherwise. Then, based on the correlations extracted by the appropriate method, missing speed data about the 3 road segments are estimated.

To increase the accuracy of traffic state estimation, they apply an information fusion technique. They adopt the *Dempster-Shafer* evidential fusion theory [136] which also enables to overcome conflicts among different input data and to compensate defective sensors. A strength of their proposal is the use of data-centric parallel computing techniques to reduce the required execution time.

This work presents the use of an MLR model coupled with a data fusion technique that improves the accuracy of traffic state estimation. It is particularly interesting as they use multiple data from vehicles to estimate missing information. This model could be generalised to estimate several different information. However, the drawbacks of this method are the execution time required and the impossibility to add new computational nodes at run time.

Regression Estimation is relevant to address macro traffic estimation problems based on prior knowledge of the variables to be estimated. However, regression is not indicated for an evolving environment because the model must be rebuilt when required, so the **dynamic** and **openness** challenges are not matched. Note that regression methods can be combined with Artificial Intelligence methods to improve results.

3.2.3 Existing Artificial Intelligence Data Estimation Methods

Artificial Intelligence (AI) includes a large spectrum of techniques addressing with success many application domains and different objectives, like data mining, objects tracking, robots decision,... Definition 2 highlights that AI can be used to understand language and learn from experience.

Definition 2. *[**Artificial Intelligence**] is the use of computer programs that have some of the qualities of the human mind, such as the ability to understand language, recognise pictures, and learn from experience. Cambridge Dictionary.*

AI has been effective to address dynamic contexts. The main idea is to use a dataset with correct answers to train the AI: this is the learning step (the previous step before the estimation). Then, during its execution, the AI chooses the best answer. An AI can process any form of information: numerical value, word, image etc., as long as the AI has been correctly trained. However, such techniques usually require a large amount of (labelled or not) data to cover all the possibilities and they require also time to train that is not always available. Naturally, AI has been widely used in the context of data estimation as it is well suited and the techniques used in this context are presented and discussed hereafter. Each method is explained with the associated strengths and limits. In the remainder of this section, the Neural Network (NN) technique is presented, then Decision Trees and Forests are described. A conclusion on the best characteristics of technique for **cross-understanding** is made.

Neural Network

Artificial Neural Network (ANN) operation is inspired from the brain neural activity and has been first proposed by McCulloch and Pitts [95] in 1943. It has been since widely used in contexts requiring a human brain like vision, tracking,... An ANN is composed of simple entities called neuron, each having a thresholding function (also called activation function) used to activate the neuron [117]. Artificial Neural Networks are built as a succession of layers: an input layer, an output layer and several ones hidden between the two. Layers are composed of many neurons, hundreds or more, depending on the application case. A neuron is connected to all neurons from adjacent layers by synapses (with an exception for the input and output layers). In an ANN, each layer is specialised in a feature. For example in facial recognition, one layer focuses on the eyes, another one on eyebrows, etc. Thanks to this decomposition, an ANN can perform complex tasks by dividing them for each layer. The input of a neuron is a weighted sum of the outputs of the neurons in the previous

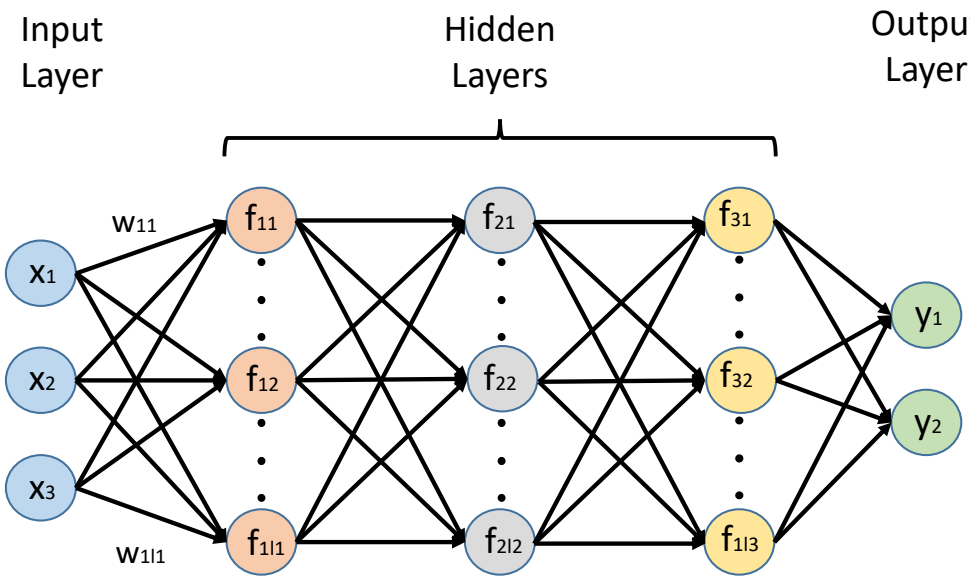


Figure 3.2: Example of a multi-layer artificial neural network with 3 hidden layers

layer. Synapses also play the role of amplifiers as they modify the neuron's output. Figure 3.2 illustrates a representation of ANN.

The first step before using an ANN is the training step. Training is the execution of the ANN on a dataset to modify synapses and threshold functions thanks to feedback given by the dataset called the *back propagation* learning procedure [127]. The principle is to adjust the connection weights to minimise the difference between the expected output and the result given by the ANN.

Some notable improvements of ANNs are Radial Basis Function Networks [110] (providing approximations to smooth functions and the learning phase is quick) and Convolutional Neural Network [36] (to obtain a better robustness in the estimation of the parameters [49]). Next are presented works using ANN to tackle data estimation problems with success.

Kumar et al. [79] have used a multiple-layer ANN to predict the traffic flow in short term. Prediction is a specific case of data estimation as the objective is to estimate the values of variables before they can be observed. This ANN is efficient to predict the traffic state within 15 minutes using traffic density, speed, time of day and time of the week. The authors highlight the limitations inherent in the black-box nature of ANN that prevents finding the interrelation between variables.

In the context of smart cities, even more in the context of smart buildings, one of the objectives is to obtain data from an area without sensors. Aliberti et al. [8] propose an ANN to estimate the temperature in a room without sensors. Their Autoregressive neural network is able, after training, to estimate accurately the temperature with data from another environment. In consequence, the ANN can apply a transformation to the available data. Unfortunately, it is highly specialised in temperature without genericity.

ANN is an efficient way to deal with the estimation of data in the environment. However, it requires an accurate training step with the dataset and unfortunately, data are not always available *a priori*. Another limitation is that they are specialised in given features, so the add of news features requires another training step. This is a limitation for the **openness** challenge. Recently, many improvements have been made to adapt the learning transfer to different contexts or models (i.e. the light modification for recognition).

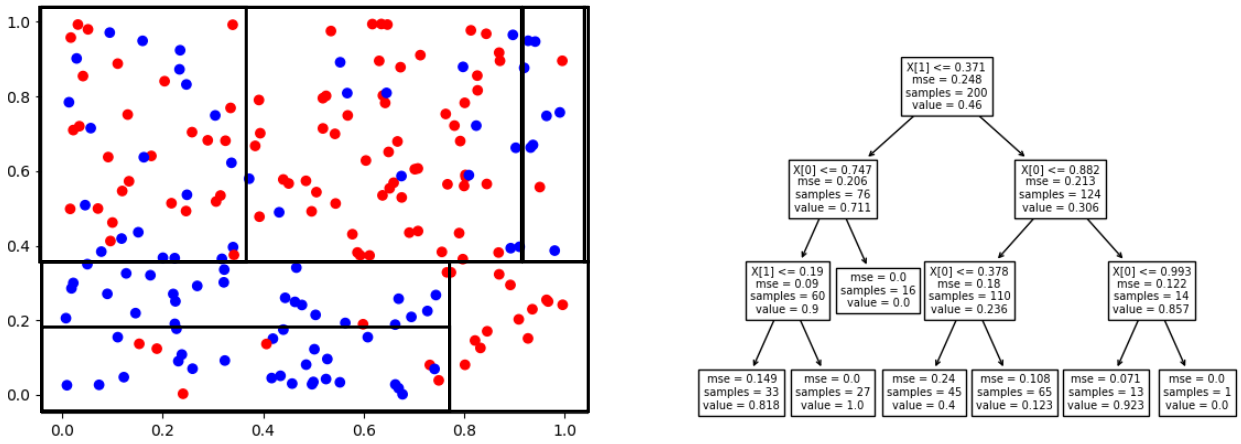


Figure 3.3: Example of a decision tree (right) and the resulting regions of the classification (left). The depth of the tree has been limited to 3 for visualisation purposes.

Decision Tree and Forest

A decision tree is a decision support tool that uses a tree-like model of decisions and the possible consequences, including the outcomes of an event, resource costs, and utility [106]. Regression decision trees are built by splitting the data to obtain accurate classes, also called regions, where data are correlated. When data is missing, it is possible to use the value of other data in the region. The estimation is the mean of all data values in the concerned region. Figure 3.3 shows an example of a decision tree using scikit learn [113] and the resulting area on the dataset.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time [62]. For regression tasks, the mean prediction of the individual trees is returned. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees [131]. A drawback of such methods is that their performance can be impacted by data characteristics.

Decision tree and tree-based models in general are unable to extrapolate learning methods to any kind of data they did not analyse before. These models are just averaging data points they have already learnt.

A Gradient Boost Decision Tree adds a boosting capability to improve results with a weak dataset. The learning procedure is called repetitively on the same dataset but each time in a different manner. By modifying the way the learning procedure is called, new results are obtained thus reducing the chance of misclassification. Zhang et al. [166] use a gradient boosting regression tree to predict the travel time on a freeway. Data are collected using heterogeneous sensors embedded in vehicles. The prediction model uses data collected and the gradient boosting regression tree completes missing values in the dataset. Thanks to the estimation, the prediction model gives an accurate result without being impacted by the incomplete dataset. To deal efficiently with heterogeneous data, this technique is computationally heavy and cannot handle too many variables.

Decision Trees and their extensions use a strategy of classification to estimate missing values. A correct estimation requires to have a correct classification of variables in regions. As the estimation is carried out with the mean of all available variables in the region, the interdependence must be accurate to avoid errors.

3.2.4 Synthesis on Cross-Understanding

Cross-understanding can be addressed according to two strategies: 1) sharing the referential frame or 2) estimating the transformation using the previous observation. Sharing the referential to merge two representations of the same environment has the major issue of being heavy in terms of communication [5]. Because the objective of this thesis is to optimise communication, the second strategy has been chosen.

Providing the capabilities to a CAV to adapt a communicated information from another referential frame to still be used by its autonomous driving process leads to the possibility to use cooperative strategies. To cope with the ITS context characteristics, a method has to be able to adapt to **dynamic** changes, and has to take into account the **openness** of the world as well as being **scalable**. As all possible information and communication methods are not likely to be known in advance, heavy learning methods may lack training datasets to adapt to every possible outcome. Moreover, storing a large amount of data inside a system with limited capacities is not a good idea. In consequence, to be as close as possible to the application case, a decentralised solution must be used to match the local control. However, to be used in the ITS context, a decentralised solution has to use local communications between entities to allow the **scalability** of the proposed solution. Looking at the state-of-the-art of adequate techniques, this is included in the data estimation domain which has been heavily studied. With the emergence of the smart city concept and ITS, such techniques give valuable information about the environment, which would not otherwise be available. In this context, standard mathematical solutions are not able to adapt to dynamic change. Moreover, the drawback of a centralised solution lies in the bottleneck of communication with too many involved entities. Indeed, if all CAVs ask a centralised controller how to transform an information, the answer may come late or not at all. Moreover, using only a local behaviour prevents the possible exchange of private information with the controller.

3.3 Optimising Communications

In today's context with the increasing number of intelligent devices, communication is a central and crucial issue. Communication has to be used wisely to avoid network saturation or bottlenecks. Thus, optimising the communication between entities appears as essential.

This section presents and explains the *Distributed Constraint Reasoning* domain. Constraint programming aims at solving decision-making problems formulated as optimisation problems of some real-world objective [125]. This domain is studied through the communication optimisation scope. First, the classical Constraint Reasoning domain is presented then a focus on Constraint Problems and Distributed Constraint Optimisation (DCOP) framework is done.

3.3.1 Complex Problem Optimisation

The optimisation domain, studied for a long time, has the objective either to increase a gain or decrease a loss for a specific problem such as minimising a plane's weight while keeping an acceptable comfort. Formally, definition 3 is used:

Definition 3 (Optimisation)**.** *Mathematical optimisation (more commonly referred to as optimisation) or mathematical programming is the selection of the best element, with regard to some criteria, from some sets of available alternatives.*

Optimisation problems arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest to mathematics for centuries.

In the simplest case, an optimisation problem consists in maximising or minimising a real function by systematically choosing the inputs values from a set of allowed values and computing the value of the function. The generalisation of optimisation theory and techniques, constituting a large area of applied mathematics, includes finding the "best available" values of an objective function given in a defined domain (or input), including a variety of different types of objective functions and different types of domains.

This section is organised as follows. First, the Constraint Reasoning domain is explained, followed by the Distributed Constraint Reasoning one, using the Distributed Constraint Optimisation (DCOP) Framework. Different algorithms classes are studied to extract the characteristics required by an algorithm to address the problem of **communication optimisation** with the properties of **scalability**, **dynamic**, **openness** and **privacy**. From this study, the two classes that can address one or several properties are highlighted and studied. Finally, a conclusion dealing with which DCOP algorithm type is the best to address the stated problem is given.

Constraint Problems

Many works and applications on optimisation, using the problem constraints, are classified as Constraint Reasoning Problem (CRP) (also known as Constraint Network) which is well-detailed in Dechter's book [30]. Common CRPs are the knapsack problem [25], the travelling salesman problem [80] and the graph colouring problem [99].

Using the Dechter's definition, a Constraint Network is a triplet $< X, D, C >$ where :

- X = $\{x_1, ..., x_i\}$ is a set of finite variables;
- D = $\{D_{x_1}, ..., D_{x_i}\}$ is the set of all finite domains and D_{x_i} = $\{v_1, ..., v_k\}$ is the domain the variable x_i takes its value from;
- C = $\{C_1, ..., C_t\}$ is the set of constraints C_j of the problem;

Constraints are functions that define a relationship among a subset of variables, called **scope** S_i, and formally, $S_i \in X$. The cardinality (the size) of a scope is called **arity**. Constraints reasoning may include *soft* as well as *hard* constraints.

A **hard constraint** limits the number of possible values combinations for variables. An example of hard constraint of the knapsack problem [25] is "the bag has a volume of 100". Thus the total volume of all the objects put in the bag cannot exceed 100. Formally, a hard constraint is a relation between the domains of the variables in its scope and is written : $C_j \subseteq \Pi_{x_c \in S_{C_j}}$, where Π is the *Cartesian* product. Consequently, it can be represented as a set of allowed assignments.

A **soft constraint** represents a preference. In the knapsack problem, a soft constraint is "if two specific objects are in the bag, the cost is increased by 2". A soft constraint is represented by a function (called utility function) that gives a *reward* (or utility) for each combination of values in its scope. A reward is a numerical value usually added to the objective function. This reward can be beneficial or detrimental to the objective. Considering the example of the knapsack problem , a reward is beneficial if the soft constraint is ensured, else detrimental (ensuring that two specifics object are in the bag, increases the gain by 5, and 0 otherwise). A soft constraint can be written as : $C_j : \Pi_{x_c \in S_{C_j}}$.

To find a solution for a CRP, a value must be assigned to each variable in X, the assignment of the solution is represented by σ. An **assignment** is **complete** if all variables have a value, otherwise, it is a **partial** assignment σ_V with V the value of all the assigned variables in the assignment σ). Solving a CRP consists in choosing the assignment such as it maximises (or minimises) the global objective function.

From constraints definitions, two problem types emerge depending on the constraints involved. When all constraints are *hard constraints*, the objective is to find an assignment where all constraints in C are satisfied.

This problem is called **Constraint Satisfaction Problem** (CSP) and the optimal value of the objective function is achieved when the assignment satisfied all the constraints (called consistent assignment). Note that in a CSP, two assignments satisfying all the constraints are equivalent. A common example of CSP is the *n-queens* problem representing a chess game with n queens. The problem objective is to place the queens without two queens being on the same row, column and diagonal.

Soft constraints enable to differentiate two assignments as they indicate a preference. The problem is no longer a CSP but a **Constraint Optimisation problem** (COP). A COP solution is a complete assignment following the next two properties:

- All *hard constraints* are satisfied (if any);
- The problem objective is either to maximise or to minimise a given function (called global objective function). Generally, the function is a simple weighted sum of all soft constraints cost.

Usually, in COP, the global objective function aggregates all the constraints cost (mainly sum functions are used). The objective is either to *maximise* or *minimise* it depending on the problem. Note, that it is possible to transform hard constraints into soft constraints by giving a reward if the assignment is not allowed; the reward, which is here detrimental, could be $+\infty$ (for minimisation) or $-\infty$ (for maximisation), and 0 if hard constraints are respected. From this statement, we can say that all CSP can be encoded into a COP. A common example of CSP, including hard constraints, is the map colouring problem. The hard constraint imposes that two adjacent nodes must not be of the same colour. The soft constraint seeks to minimise the total number of different colours used in the entire map.

$$F_g = \begin{cases} argmin \sum_{C_i \in C} C_i & \text{if minimisation} \\ argmax \sum_{C_i \in C} C_i & \text{if maximisation} \end{cases} \tag{3.2}$$

Constraint Reasoning Problems Extensions

Among the many CRP extensions studied these recent years, two extensions (Dynamic Constraint Reasoning and Distributed COP) can address the stated challenges of dynamicity and scalability (exposed in chapter 2) and are described hereafter.

Dynamic Constraint Reasoning addresses problems with the time dimension. The stated problem is dynamic as its variables and constraints evolve (over time). Addressing such problems requires to find a solution in constrained time.

Distributed COP tackles problems that are distributed among several entities. The main objective is to subdivide the problem into several local subproblems. Doing so, it facilitates the resolution of a large problem with a trade-off on the result. To better understand the CRP domain, and more particularly the COP one, an example of COP formalisation on the well-known optimisation problem, the k-colouring problem, [27] is presented. The objective is to colour each node of a graph with maximum k colours where the colour of a node must be different from the adjacent nodes. Figure 3.4 shows an example of a k-colouring problem with six nodes. In addition, there is a preference on which colour is used. In this example, the maximal number of colours possible is set to 4:

The triplet of this problem with $k = 4$ is:

- $X = \{A, B, C, D, E, F\}$.
- $D = \{1, 2, 3, 4\}$ Each number represents a colour (1 = green, 2 = yellow, 3 = blue and 4 = red).
- $C = \{A \neq B, A \neq C, B \neq C, B \neq D, C \neq E, D \neq F, E \neq F \sum_{i=1} x_i^2\}$.

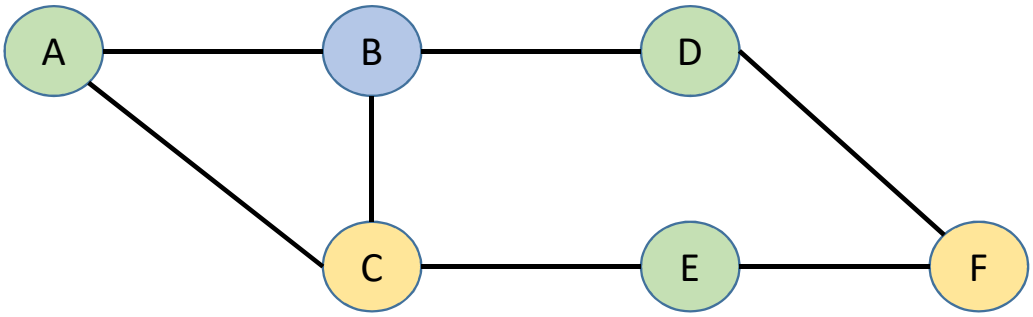


Figure 3.4: Example of a 4-colouring problem.

This problem is a COP as the last constraint is a soft constraint. The objective is to minimise the total cost by choosing the colour of each node with a preference order (green better than yellow, yellow better than blue and blue better than red). A solution of this problem is: $A = 1, D = 1, B = 3, C = 2, E = 1, G = 1, F = 2$. This assignment is complete as all variables have a value and hard constraints are respected. Moreover, only 3 colours have been used, thus the solution is better than a solution with 4 colours.

3.3.2 Distributed Constraint Optimisation Problem

DCOP is a powerful framework to address COP in a distributed way [39]. The problem and variables are distributed among entities called agents. In addition to the triplet from the CRP formalisation, two fields are added: the set of agents owning a part of the problem and the function linking agents and variables. Formally, a DCOP is:

- A = $\{a_1, a_2, ..., a_m\}$ is the set of agents;
- X = $\{x_1, ..., x_i\}$ is a set of finite variables;
- D = $\{D_{x_1}, ..., D_{x_i}\}$ is the set of finite domains and D_{x_i} = $\{v_1, ..., v_k\}$ the domain the variable x_i takes its value from;
- F = $\{f_1, ..., f_t\}$ is the set of cost functions (soft constraints) of the problem. Note that in DCOP, all the constraints are represented as soft constraints, thus the set of constraints is replaced with the set of cost functions. In DCOP all constraints are soft constraints but a hard constraint can always be transformed into a soft constraint. Thus there is no loss in generality;
- α : X $\rightarrow$ A is a total and onto function from variables to agents and it assigns the control of each variable x $\in$ X to an agent $a(x)$;

The agents in the formalisation enable naturally to analyse this domain as a part of the Multi-Agent System paradigm. As with COP, a DCOP solution is a complete assignment of all the variables that optimises a global function. The main difference between COP and DCOP lies in the fact that computation and knowledge are distributed among agents. Without losing in genericity, a *hard constraint* is a soft constraint with an infinite cost.

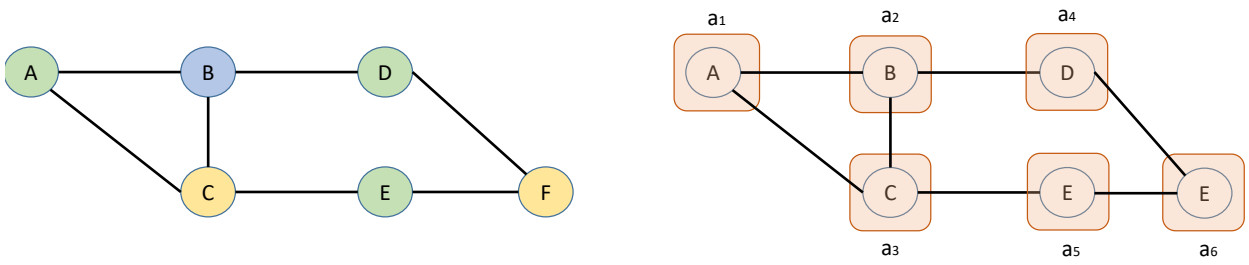


Figure 3.5: Graphical representation of the map colouring problem.

DCOP Graphical Representation

Graphical representation gives a comprehensible illustration of problems (CSP, DCSP and DCOP). One common way to represent a DCOP problem is the constraint graph. Note that those graphs can be used for COP but the focus is on DCOP. In such graphs, circles represent variables and rectangle constraints (functions). Each edge between a variable and a constraint means that the scope of the function includes the variable. The graphical model of a DCOP adds to the graph the agent's field. Variables and functions are enclosed by agents, represented by an orange area (every node in the area belongs to the agent). The easiest example is the map colouring problem as it is already a graph 3.5 (for visualisation purposes, constraints are not represented). On the left, a graphical representation of the problem is given, and on the right there is an equivalent DCOP one. Each node is controlled by an agent which decides the node value (in this case, a colour).

DCOP Extensions

Fioretto et al. [39] present the five major extensions of the DCOP framework:

1. **Asymmetric DCOP (ADCOP)** addresses problems with local preferences and desires [50]. In classical DCOP, the same constraint has the same cost for every agent affected by it whereas, in ADCOP, each agent has its own goal and its own representation of the environment. Such problems are difficult to solve as the local preferences of an agent may not be those desired by its neighbours. One of this extension main challenges is to optimise a problem while respecting the privacy of agents, thus adequate for this thesis challenge. Note that it is always possible to transform an ADCOP into a classical DCOP by introducing mirrors variables. Many algorithms have been developed to tackle ADCOP problems with increased efficiency [89] [51] [148].

2. **Dynamic DCOP (Dyn-DCOP)** models a problem that evolves with time [129]. Usually, the solution of a DCOP is not (and does not need to be) modified. In a real-world application, especially with IoT and IoV, the environment is not static. Dyn-DCOP enables to model correctly such problems because it adds to the problem the time dimension that enables constraints, variables and agents to be modified continuously. The goal is to find a reliable solution before a modification is observed thus making the previous optimisation obsolete. There are two ways to address such problems. The first one is to model the problem as a sequence of classical DCOPs with a time constraint to find a solution. The second is to use previous assignments, without restarting from scratch.

3. **Probabilistic DCOP (P-DCOP)** models problems in which agents must discover the constraints during the optimisation process [162]. The cost of the functions is at first incomplete or unknown. This knowledge is collected while agents choose values for their variables. P-DCOP is used to cope with real-world issues, also called perturbations, such as weather conditions or malfunctioning devices.

These exogenous perturbations modify the agents' behaviour. This is modelled by cost functions that are stochastic.

4. **Multi-Objective DCOP (MO-DCOP)** models problems with not one but several global functions that contradict each other [156]. The purpose is to balance objectives to satisfy every agent without having one not satisfied. Formally, the objective is to find an assignment ensuring the **Pareto optimality**. "Pareto efficiency or Pareto optimality is a situation where no individual or preference criterion can be better off without making at least one individual or preference criterion worse off or without any loss thereof". Examples of such applications are discrete calibration, economic and feature selection.

5. **Quantified DCOP (Q-DCOP)** models the problem that includes competitive agents with antagonist goals. Examples are competitive games, planning and verification [94]. Uncooperative variables in such problems are quantified and can be *universal* (prefix $\forall$) or *existential* (prefix $\exists$). A universal variable is unassigned by the optimisation process while an existential one has a value from its domain. Therefore, a Q-DCOP has no optimal cost but a lower and an upper bound on the solution cost. The solution is acceptable when the solution is within the bounds.

Among all these extensions, 2 are particularly interesting in the context of this thesis because they address the stated challenges. Indeed, the ADCOP one preserves the local knowledge of agents thus keeping the personal information **private**. The Dyn-DCOP can address problems with high **dynamic** changes. Thus, both extensions are studied later with a description of the resolution algorithms. To accurately describe the algorithms, the next section presents all the characteristics to take into account when addressing a particular problem.

DCOP Taxonomy

DCOP algorithms are classified using several characteristics. [39] propose a taxonomy of classical DCOP algorithms, which is illustrated in figure 3.6. This taxonomy helps to identify the algorithm best suited to the challenges related to dynamics, openness, privacy and scalability.

Completeness / Incompleteness: completeness criteria classify algorithms according to whether or not they guarantee to find and return an assignment optimising the global objective function F_G. Such an algorithm is called *complete*. ADOPT [23], DPOP [41], SyncBB [147], pC_DPOP [115] and OptAPO [88] are some of such well-known algorithms. However, those algorithms are computationally expensive as they explore a large part of the space. *Incomplete algorithms* are faster with a lower memory burden but the resulting quality is impacted. These algorithms do not guarantee an optimal assignment but a **near-optimal** one. They are divided into 3 subclasses: **approximate algorithms, bounded-approximate algorithms and region optimal approximate algorithms**. Approximate algorithms that do not guarantee bounds are: DSA [86], Max-Sum [24] and MGM [86]. The bounded-approximate algorithms guarantee that the assignment gives a solution within bounds. This means that the found solution cannot exceed a distance that is given as input. Bounded-approximate algorithms are Bounded ADOPT [114], Bounded Max-Sum [124], D-GIBBS [108] and DUCT [112]. Region optimal approximate algorithms are near complete algorithms as they enable to find an optimal solution for specific regions that are given by the user. To decide the *amount* of completeness needed to address a problem, one must know the balance between speed and solution optimality.

Centralisation / Decentralisation: the centralisation criterion focuses on the independent behaviour of agents to make a decision. Usually, DCOP is fully decentralised with only local behaviours but some algorithms take advantage of coordination between agents in the optimisation process with a gain of performance. The algorithm is so classified as **partially centralised**. One drawback of partial centralisation is a loss of privacy among agents, possibly making some private information available to others. OptAPO and PC-DPOP

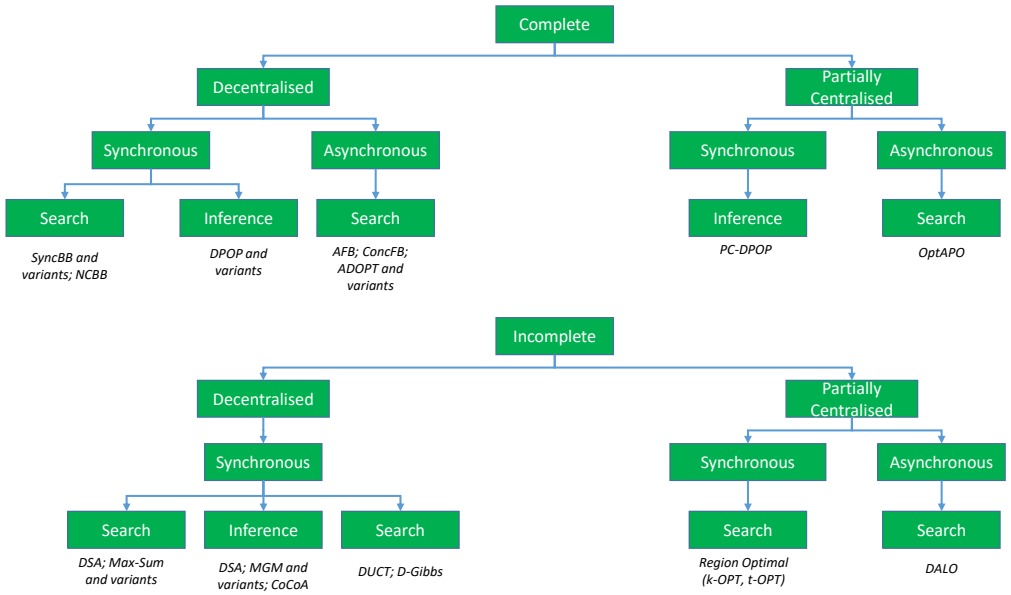


Figure 3.6: Taxonomy of DCOP algorithms

are partially centralised DCOP algorithms. Note that in partial centralised DCOP, message exchange (time and cost) for coordination are taken into account and can be damageable for the application. The motivation of using centralisation relies on the necessity for agents to keep private their personal information.

Synchronisation / Asynchronisation: synchronisation is related to how agents update their local information. In a **Synchronous** DCOP, agents wait for each other before going further in the optimisation process following an order. In **Asynchronous** DCOP agents tend to be faster at modifying their variables and their local behaviour as they do not need to wait for each other. The drawback is that their knowledge can be outdated as their representation is no longer true because of one neighbour's decision that has modified it. It has been shown that asynchronous agents may affect the network load and performance. The choice of synchronisation depends on the problem nature and to what extends agents decisions have an impact on others.

Exploration Strategy: Yeoh [164] classified the **exploration** process of DCOP algorithms into three different classes: **Search**, **Inference** and **Sampling**.

- The **Search** approach can be divided into two sub-classes: **complete search** and **local search**. In low dimensions (low number of agents, variables and function), the complete search is efficient as it explores all the possible assignments until finding the best one. Many works on this approach focus on reducing space exploration by removing a subset identified as not containing an optimal solution. However, even with those techniques, a complete search is not efficient for high dimensional problems. **Local search** emphasises the local knowledge and neighbourhood of agents. Agents reason only on a restrained part of the environment and improve their solution thanks to the interactions with their neighbours. Solutions scale well and are light computation even with high dimensional problems. However, those solutions are often stuck in a local optimum without the possibility to get out. Most of the time, the solution found is near-optimal. The gain in speed of the local search is counterbalanced by the loss of optimality of the solution, which must be acceptable for the addressed problem.

- The **inference approach** is related to the sharing of knowledge between agents. Agents propagate their influence (the effect they have on the problem) to other agents in the graph until enough knowledge is obtained. Then agents decide variables they own. By doing this, the problem is reduced and more accessible.
- **Sample approaches** approximate a function (typically a probability distribution) as a product of statistical inference to sample the search space. Note that this approach only exists for incomplete algorithms.

Similarly to the completeness criteria, choosing an exploration approach is related to the problem dimension and quality of the researched solution.

Conclusion on DCOP Solutions Characteristics

DCOP is a powerful framework able to model a large variety of different problems. In this section, the main characteristics of DCOPs have been presented and the four main criteria (completeness, centralisation, synchronicity and research method) to categorise a DCOP algorithm. From the study of DCOP extensions, two possess efficient characteristics for the communication optimisation challenge: ADCOP and Dyn-DCOP. The next section describes more precisely the two main extensions addressed in this thesis.

3.3.3 ADCOP Analysis

Asymmetric DCOP enables the agents' local preferences modelling. An ADCOP example is presented in table 3.1. The possible assignment of Agent A_1 (respectively A_2) is α or β (respectively γ and δ). The costs for A_1 correspond to the left-side of cells (respectively right-side for A_2). So, when A_1 chooses α for value and A_2 chooses δ for value, then the resulting cost is 3,1. In this example (with a maximisation objective), A_1 prefers to assign β but α would give a better cost for A_2.

A_1/A_2	γ	δ
α	5,9	3,1
β	6,2	4,7

Table 3.1: Example of an ADCOP with two variables and two agents.

ADCOP is especially relevant in the context of Smart Cities, IoT and IoV. User preferences may be private information. Thus, exchanging that information, even for optimisation purposes, is not desirable. One of the main objectives is to lower the *privacy* loss. In ADCOP, privacy refers to the variables and functions owned by agents. However, the scope (S_i) of the agent's functions may include variables owned by its neighbours. Usually, agents do not know the impact of their variables on other agents. Looking at the example in 3.1, agent A_1 does not know the effect of the value α for A_2.

ADCOP has been used by Maliah et al. [89] to model an economic problem of collaboration in a company. They present a privacy-preserving planning algorithm in which the agents collaboratively generate an abstract and approximate global coordination plan and then individually extend the global plan to executable plans.

Next section presents ADCOP algorithms with characteristics that are in this thesis scope.

Maximum Gain Message (MGM)

MGM is an algorithm that is incomplete, decentralised, synchronous and uses a local search [86]. Historically, MGM is a modification of the DBA (Distributed Breakout Algorithm) [61] to improve messages exchange.

The idea of MGM is that, at each cycle, each agent chooses the best assignment according to its local problem. An agent effectively assigns a variable if the resulting gain obtained is better than the cost gain of all its neighbours. This algorithm is divided into 5 main steps.

- First each agent chooses a random value (from the variable domain) for each of its variables ❶. The next steps are made while the optimisation process is not over;
- Each agent shares values of its variables with its neighbours ❷;
- Then, each agent computes the best new assignment it can do according to its local problem and sends the potential cost gain to its neighbours ❸;
- An agent decides to modify its assignment if its gain improves its solution and if the gain is better than all its neighbours' (ties are broken using indexes) ❹;
- The last step requires a synchronisation of all agents ❺. The stop condition of MGM can be either a time limit given in input or a gain too small;

In classical DCOP, MGM ensures an improvement monotony of the solution. The solution can only improve with each cycle, which is an expected characteristic. However, as MGM is a local search algorithm, it may be stuck at a local optimum. MGM was improved by including the possibility to coordinate several agents together by using coalitions. Those extensions are called *MGM-k*, with the most known one: MGM-2 (coalitions have a maximum of two agents). It has been shown that MGM-2 provides better results even if it is a partially centralised approach.

The monotony is broken in the case of ADCOP because the gain from an agent can lead to a worse loss for another agent. For example, an agent computes a potential cost gain of 5 by changing the value of its variable "x", this gain is the best among agent for the cycle ❸. But when effectively changing its variable x value ❹, the global cost is reduced. This happens because another agent loses 6 with the new x value. To cope with this issue, two extensions of the MGM have been proposed to address ADCOP specifically.

Minimal Constraint Sharing (MCS-) and Guaranteed Convergence Asymmetric (GCA-) MGM

The Minimal Constraint Sharing MGM has been proposed by Grinshpoun et al. [51] to improve the MGM algorithm for ADCOP. The main idea behind MCS-MGM is to add the possibility for agents to share their constraints with their neighbours. MCS-MGM follows the same steps as the classical MGM until the step of sharing the values, referred as ❷. So, after collecting values ❷, each agent evaluates the consequence of its neighbours' actions on its local problem ❻. For each neighbour whose action has resulted in a worse loss than the proposed gain, the agent sends its side of the constraint and puts it to 0 ❼. Then, each agent gathers the updated version of the constraint that modifies the local sub-problem ❽. Finally, the algorithm follows the standard MGM from the step ❸.

The exchange of constraints allows agents to increase their knowledge, thus making it possible to choose a better assignment. MCS-MGM, therefore, gives better results than MGM (and MGM-2). However, this has a cost both in time and in privacy loss (note that MGM-2 has almost a 100% privacy loss). Considering the time computation, the step ❽ is a synchronisation stage where the agents wait until the other agents have made their decision. The algorithm has two synchronisation steps, ❽ and ❺ which, as stated before, induce a possible time loss. Authors, Grishpoun et al. [51] state that the privacy loss is low as only a small part of each agent structure is shared. However, this may be some sensitive information that a real case application cannot envisage sharing.

The Guaranteed Convergence Asymmetric MGM (GCA-MGM), similar to MCS-MGM, modifies the condition of sharing a constraint [7]. An agent shares its side of the constraint if it has encountered a loss of gain, even if it is less than the predicted gain of its neighbour. Consequently, more constraints are exchanged than in MCS-MGM thus leading to more privacy loss. However, GCA-MGM is guaranteed to converge. An observation of the results of MCS-MGM and GCA-MGM highlights that MCS solutions are more often better than GCA-MGM. Indeed in GCA, some constraints exchanged may lead to fall into a local optimum, whereas MCS would not.

Both algorithms are efficient to address ADCOP with an improvement both in quality of the result and on privacy compared to MGM-2, even if the added synchronisation phase makes it slower than standard MGM.

Synchronous Branch-and-Bound (SyncBB) and its Extensions

Synchronous Branch-and-Bound (SyncBB) [60] is a complete, decentralised, synchronous and search algorithm where agents are statically ordered. It has been designed for common DCOPs. One after the other, each agent computes a partial assignment which is then given to the next agent that will also calculate the partial assignment and give it to its successor. This is repeated until the cost is lower than the best solution already stored or the last agent is reached. Then a *backtracking* procedure is used to continue to explore the research space. The optimisation process ends when all the space is explored or *pruned* to avoid a complete exploration.

SyncBB has been extended to address ADCOP, called SyncABB-2ph by Grishpoun et al. [51]. This algorithm follows a two-phase strategy. The first phase is similar to classical SyncBB and consists in finding a complete assignment. Phase 2, which begins when a complete assignment is found, consists of all agents aggregating the missing cost that was not considered. Then it ensures that the cost of the assignment is not worse than the bound corresponding to the best solution already found. In that case, the algorithm restarts to step 1 with a backtracking procedure.

A focus is done on two extensions, P-SyncBB (Privacy-Preserving SyncBB) [52] and PC-SyncBB (Privacy-Preserving Collusion Secure) [148], with a particular emphasis on the privacy criteria. P-SyncBB restrains agents knowledge by only giving the upper bound to the first agent of the sequence to know the upper bound. This is authorised because, in SyncBB, an agent never knows the assignment computed by the following agents of the sequence. As it is the first of the sequence, it does not know any other assignment except its own. Following agents need to request the first agent to compare the current cost to the bound. A recent extension [148] enables to be secure enough to coalitions of agents by implementing a cryptographic weaponry. As efficient as they are, cryptographic processes are time-consuming and require agents' synchronisation.

SyncBB and its extensions are efficient but the algorithm **completeness** makes it not suitable to address Dyn-DCOP problematics. However, the addition of a secure protocol is interesting as it ensures privacy.

Synthesis on ADCOP

ADCOP is used to model problems where agents have personal preferences over a solution. **Privacy** is the main concern in many real-world application cases to secure sensitive data. In ADCOP the exchange of private information is a way to improve the optimisation process but must be avoided. What is gained in result by making its preferences available to others, is lost in privacy. Moreover, when addressing ADCOP, one must be sure of which information can be shared or not.

One of the presented works shows that it is possible to add security mechanisms to protect private information, with a cost on computation time. The use of cryptographic mechanisms, hiding the real information behind it, is a robust way to ensure security and privacy. In the ITS context, security is a key

issue in relation to the danger posed by a hacker. However, despite its importance, this point is not discussed in this thesis. An assumption is made that there are no malevolent agents that may disturb the overall system.

In this section, the challenge of **privacy** has been studied with the ADCOP extension. **Scalability** can be addressed by selecting the correct space exploration method. In the objective of communication optimisation, two challenges remain: **dynamici** and **openness**. The next section studies the Dyn-DCOP which is a solution to the these two challenges.

3.3.4 Dynamic Constraint Optimisation Problem

Dynamic Constraint Optimisation Problem is a subset of COPs [157]. In such optimisation, the problem is susceptible to being modified once it is solved or during the optimising process. The dynamics has to be taken into account to solve many real-life applications, as in Intelligent Transport System. Entities (CAVs) this thesis focuses on are mobile with a not fixed communication range (because of obstacles). From one CAV's point of view, the environment and the neighbourhood are highly dynamic and environmental information are changing. In addition to the dynamics, openness needs to be taken into account. Optimising communication in a fleet of CAVs is a continuous process in time. Taking into account the time dimension in COPs adds a major challenge: is the optimisation process fast enough to end (i.e. to find a solution) before new changes occur? With this in mind, fast solutions must be preferred to slow ones. However, some methods anticipate changes to better adapt to them. This section first presents and illustrates the particularity of Dyn-DCOP as well as two types of approaches.

The Dynamic DCOP (Dyn-DCOP) extension

Dynamic DCOP (Dyn-DCOP) models problems that are dynamic (not static) in time. Constraints, variables and agents can enter, leave or be modified during execution. Dyn-DCOP is efficient for many real-world applications like IoT and IoV where a never-ending optimisation is needed. An example is the optimisation of a user's connection (via its smartphone) to a station. Several users leave or enter the system in an unexpected way and stations may be saturated. The objective of Dyn-DCOP is to be able to adapt to changes over time. Two methods are then possible: 1) the optimisation process is fast enough to find a solution before the occurrence of a modification or 2) modifications can be anticipated so that the optimisation process does not have to restart from scratch at each event.

Rust [129] has designed a Dyn-DCOP algorithm that allocates computation among several devices, in the context of IoT. This distribution of computation allows solving complex tasks. However, as those devices are susceptible to failures, it may lead to missing computation. The author's objective is to design a resilient algorithm that can deal with the arrival and departure of agents. The strategy is first to replicate a computation into several different devices, so that the loss of a part of the computation is avoided. After proposing a replicate method, he proposes a cooperative procedure to fairly distribute replicates among several devices to avoid having one with too much computation. Thus, this algorithm achieves the objective of resilience and takes into account the communication cost.

Dyn-DCOP may be modeled by creating a sequence of classical DCOP $\{P_1, P_2, ..., P_{tmax}\}$ where P_t is a classical DCOP. In such a sequence, P_t is the consequence of modifications encountered after P_{t-1}. This representation is efficient for a problem that can be anticipated with known changes and on which sufficiently fast classical algorithms can be used to address (e.g. incomplete and local search algorithms are fast by nature). Several algorithms have been specifically designed to address such problems and are classified according to two main approaches:

- The **Proactive Approach** [63] assumes that modifications are known or, at least, can be partially anticipated. Formally the set of transition functions [64] is accessible at P_t for P_{t+1}. Proactive

approaches anticipate robustness criteria for the solution they find. The robustness of a solution is evaluated by the number of assignment changes required after the problem modifications. Each P_t of the sequence can be seen as a Multi-Objective DCOP with the two objectives: 1) solution quality and 2) robustness. Current algorithms that implement a proactive approach are either offline or limited in scalability with a few agents only.

- The **Reactive Approach**, the most studied approach for Dyn-DCOP aims at solving the static DCOP of the sequence before a modification occurs thus leading to a new problem to solve. Thus each DCOP is solved sequentially, meaning that classical algorithms can be theoretically used on this extension. An important constraint is added; it concerns algorithms that must be able to give an assignment (near-optimal at least) faster than the environment changes. If the change occurs before a solution has been found, the algorithm continues to solve the problem which is now considered useless. Moreover, it is also possible that it may never give a solution if it encounters a new problem before an assignment for each variable is decided. Two variants have been proposed to specifically address those characteristics and constraints. The first variant uses the knowledge obtained with the preceding resolutions to speed up space exploration. Assuming that space is not entirely modified at each new problem (which is the case in most real case applications), this solution would not restart from scratch. The second variant uses algorithms that can adapt through changes while solving problems (this one continuously evolving) that have also been proposed. It may be required to include the cost of switching the solution in the problem.

Dyn-DCOP is still an active research domain, especially for dynamic and critical systems. Time is the main constraint that limits efficiency of classical algorithms. Only fast algorithms can be considered, and according to the taxonomy illustrated in fig.3.6, the fastest algorithms are the incomplete and local search ones. A solution to cope with dynamic perturbations is to add an adaptive process enabling one to continue the research without having to restart from scratch.

3.3.5 Conclusion on DCOP Framework

DCOP is a powerful framework to tackle optimisation problems that can be distributed. Taking advantage of the Multi-Agent paradigm to solve a complex problem, many works and solutions have been proposed and implemented to solve DCOPs. Looking closely at characteristics of **scalability**, **dynamic** and **openness**, incomplete and local search algorithms are to be preferred. Existing algorithms (MGM extensions and P-SyncBB) on ADCOP are unfortunately time-consuming, or privacy leaking. MGM extensions share personal information to improve the knowledge of other agents thus not matching the **privacy** requirement. P-SyncBB does not scale well due to the completeness of the algorithm. However, inspired by their architecture, a proposition is to find a mean to share efficiently agents' goals without revealing their constraints as they are. The table 3.2 summarises the quality of each presented algorithm according to the criteria of **asymmetric**, **dynamic**, **openness**, **privacy**, **scalability**, and **result quality**. Currently, no solution matches all criteria with an optimal result. The optimal solutions have difficulties to scale-up and to cope with unexpected events. Indeed, the proactive approach requires knowing in advance the changes that will happen. Asymmetric algorithms are efficient toward asymmetric problems but the loss of privacy is problematically excepted for P-SyncBB which fails to scale up.

The implementation languages of DCOP algorithms influence the computation speed, which can be a problem for the evaluation of methods. The library pyDCOP [130] offers several DCOP algorithms with a common implementation. Developed in Python, the library provides several DCOP algorithms with requirements depending on the problem being addressed. Thus, in a non-exhaustive way, a user can specify the owner of variables, the computation cost, the problem modifications for Dyn-DCOP and much more.

Technique	Asymmetric	Dynamic	Openness	Privacy	Scalability	Result Quality
MGM	–	-	+	-	+	Near-Optimal
MGM2	–	-	+	- -	-	Near-Optimal
MCS-MGM	++	-	+	+	-	Near-Optimal
GCA-MGM	++	-	+	-	-	Near-Optimal
P-SyncBB	++	- -	-	++	- -	Optimal
Proactive Approach	–	++	+	-	–	Optimal
Reactive Approach	–	++	++	-	+	Near-Optimal
AMAS	+	++	++	++	++	Near-Optimal

Table 3.2: Synthesis of presented DCOP algorithms

3.4 Conclusion on Usual Methods

This chapter has presented the state-of-the-art techniques of **data estimation** to address the **cross-understanding** objective. **Data estimation** aims at giving tools to estimate the value of information when this one is unavailable. This can be applied in a wide variety of domains. In this thesis, the context of fleet of CAVs is focused on. Indeed, sensors may be subject to failures or mostly to obstacles that limit their perception. A CAV can communicate to other CAVs information they perceive thanks to V2V communication. By doing so, a fleet of CAVs is theoretically safe from to missing information. In Besides the communication itself, sharing information is subject to the CAV's referential frame. The formalisation of the referential frame transformation into another makes it addressable by data estimation methods. In consequence, a study of the state-of-the-art of data estimation domain is provided. After explaining usual methods with their strengths and weaknesses no technique matches the four challenges of **dynamic**, **openness**, **scalability** and **privacy**.

Looking at the second challenge concerning the **overloading** of other entities, a CAV needs to optimise its communication volume to only share the most useful information with others. This can be formally represented as a Constraint Optimisation problem. Moreover, the natural distribution of entities leads to the use of the Distributed Constraint Optimisation framework. This chapter provides a study of the DCOP framework state-of-the-art methods with a close look at two extensions: Dynamic DCOP and Asymmetric DCOP. The first one is related to the challenges of **openness** and **dynamic** and the second one tp the challenge of **privacy**. Its decisions may be subject to a lot of private information that must not be shared. This can be information from the constructor's autonomous driving process which is likely to be private, or even from passenger ones like destination or driving habit. None of the presented methods is **dynamic** as well as **asymmetric**.

This work on the state-of-the-art shows, there is a need to give to a CAV two additional modules providing it with social skills. As standards techniques do not match all characteristics of **scalability**, **dynamic**, **openness** and **privacy**, this thesis proposes two Adaptive Multi-Agent System (AMAS) based modules to address them. The first module addresses the **cross-understanding** problem. The second one addresses the **communication optimisation** problem. Indeed, the AMAS approach matches the DCOP characteristic of incompleteness and decentralisation that an algorithm must follow to achieve both **scalability** and **privacy**. Moreover, the Dyn-DCOP extension states that one way to cope with dynamic and openness is to adapt the solution when perturbations are encountered in a short time. These modules have to be as light as possible resources and computational points of view. The next section presents the AMAS approach that is compliant with all stated challenges.

3.5 Adaptive Multi-Agent System

The Intelligent Transport System is a wide complex system that is in constant evolution. More and more vehicles are equipped with intelligent technologies to perceive the environment but also to communicate. As an entity evolving with other social entities (machines but also humans), a CAV is considered a social entity. Social interactions are governed by codes that CAVs must follow. Another possibility for CAVs is to create their social code, more accurate to their physiology without being limited by human social code. Looking at the evolution of communication over human history, social codes emerge from the continued interaction between humans. This thesis objective is to study how CAV can efficiently communicate with each other. A focus is done on the **utility** of communicated information to help other CAV.

The idea of a social brain proposed by Loke [85] allows considering an entire part of the CAV dedicated to managing the interaction with other road users. Formally, the fleet of CAVs is considered as a complex system which is defined in the definition 4, proposed by Holland et al. [65].

Definition 4. ***[Complex System]*** *A complex system is a system that is difficult/impossible to understand all the causes of phenomenons that happen and could happen. Phenomenons are the result of interactions of the multiple entities composing the system. Such a phenomenon is called* ***emergence****.*

The AMAS approach is efficient towards problems with dynamic, openness, scalability, and privacy. It has been applied with success on both **data estimation** [53] and **optimisation** [70], but with other hypotheses.

This section aims at presenting the Adaptive Multi-Agent System approach which is an extension of the Multi-Agent System approach. First, an introduction to complex systems and the emergent phenomenon is given, to better understand of the concept. Then it is followed by the formal description of the MAS approach and its bottom-up design. Finally, a presentation of the AMAS extension is given to motivate its use in the fleet of CAVs context.

3.5.1 The Sum of Its Part is Not Equal to the Complex System

To illustrate what is a complex system Holland et al. [65] uses the rain forest system as an example. "*The rainforest contains an almost endless variety of species—one can walk a hundred paces without seeing the same species of trees twice, and a single tree may host over a thousand distinct species of insects. The interactions between these species range from extreme generalists ('army' ants will consume almost anything living in their path) to extreme specialists (Darwin's 'comet orchid', with a foot-long nectar tube, can only be pollinated by a particular moth with a foot-long proboscis—neither would survive without the other)*". The parallel with fleet of CAVs is justified by the fact that the number of vehicles, infrastructures, and observations is too large to be understood in its entirety. Road traffic in an urban area is the result of all micro-decisions of each road user.

Looking back at the definition 4 from Holland, a complex system is composed of so many entities and characteristics that it is impossible to observe all phenomena happening at the same time. Each actor evolves within this environment and modifies it by his actions: the result of an action changes the environment in which it is done. These changes influence the action of another actor then creates modifications producing a bigger phenomenon that **emerged** at the system level.

Emergence is naturally occurring and observable in many real systems. A flock of birds or a school of fish are good examples. Fishes have a simple behaviour that consists in staying at a certain distance from their neighbours. When a predator comes, fishes start to flee away from it. Because the distance shortens, the neighbours fishes move away from the predator without needing to see it. Using this simple behaviour, the system of a school of fish is efficient to avoid predators. In this example, a fish is an actor in an environment composed of water, other fishes in the school and predators. A fish action is to move inside the water and the

fish follows a simple rule that is "keeping an adequate distance with other fishes". When a fish decides to move it disturbs its neighbours, which at their turn move and it continues on and on. In consequence, when a predator comes, fishes that can see the threat move away from it, which encourage others to do so, creating a group avoidance mechanism for the school of fish, **emerging** from all fishes' actions. As one of the most easy to understand complex systems, this illustrates the concept of emergence which is a phenomenon resulting from the behaviour of entities composing the system, often those entities are not aware of such phenomenon happening.

The study of complex systems has a grand interest in many different fields, from natural ecosystems like ants and termites to artificial system like smart cities and fleets of CAVs. It is motivated by a will to study a system in its entirety instead of its parts. In other words, it looks at complex phenomena created from the interactions of the parts between them. Smart cities and Intelligent Transport Systems are complex systems, justified by:

- The system is **open**, the **number** of actors in the system is continuously evolving;
- The control decentralisation among all those actors. It means that the control is at the micro-level, as a macro-level control is impossible;
- The heterogeneity of all entities. Connected and non-connected, autonomous and non-autonomous vehicles, people, etc;
- Several phenomena emerge from drivers' behaviours like traffic jams. Many of them are caused by multiple actions from aggressive driving, road choice or not being attentive.

In Computer Science, the Multi-agent System approach is inspired by complex system characteristics to address them. Instead of a top-down design process, a bottom-up process is preferred. A MAS is divided into two parts: the **environment** and the **agents**. Agents evolve within the environment with their local behaviour and they have a local representation of the environment. This means an agent perceives a subset of the environment that depends on its perception capacities (eyes, sensors, etc). Its environment representation is composed of the knowledge it has about it, that may be incomplete depending on its perceptions. To reflect the example of the school of fishes, agents are the fishes and the environment is the water. A fish sees in a short distance and its environment representation is composed of other fishes and predators. Depending on its position, a fish can think that there is no predator while another thinks the opposite. The two fishes have two different local representations of the same environment.

In some MAS, the solution is not encoded within the MAS but comes from agents' behaviour and is called *emergent*. A definition of emergence in computer science is proposed by [133],

Definition 5. *"An emergent phenomenon produced by a software is an interpretation of an attractor the system has converged into, which is practically unpredictable given the functionality of system component."*

The composition and interactions of micro-events affect macro observation. This micro/macro relation is the essence of MAS, focusing on the design at the micro-level, the macro-level is modified. Usually, in MAS, the macro-level represents the global objective of the system (i.e. dodging a predator for the school of fishes). This duality of micro/macro has to be taken into account in MAS. In the ITS context, MAS has many benefits:

- Natural and intuitive problem solving by active entities;
- Autonomous agents are efficient to model heterogeneous systems;
- High-level abstraction can be used to describe agents and their interactions;
- Agents are flexible and able to adapt; an agent models a real entity allowing to include all problem-solving phases [15].

3.5.2 Multi-Agent System

Multi-Agent Systems (MAS) are artificial systems composed of multiple autonomous artificial entities called agents. A MAS design follows a bottom-up approach, which means it mainly focuses on the agent design and collective behaviours. So, MAS are efficient for naturally distributed applications [47]. These applications are divided into three main classes: 1) *spatial distribution* where agents are spatially located; 2) functional distribution where skills are distributed among several agents and 3) semantic distribution when vocabularies and ontologies are different. In this section, the studied system is the Intelligent Transport System one and is used as an example to illustrate the MAS characteristics. This section begins with the description of the entities called agents. This is followed by the description of the environment in which agents evolve. After, the concept of *self-organisation* in MAS is explained. Finally, common assumptions and limits are discussed.

The Agent

An agent is a numerical entity evolving in an environment, and can represent a robot, a cell or even a mathematical function operand. A commonly used definition of agent is given by Weiss [159] (definition 6).

Definition 6. *"an agent is a computer system that is situated in some environments, and that is capable of autonomous actions in this environment to meet its design objectives."*

An agent is autonomous by nature, evolves in an environment where it has a local perception, and has its personal goals [37]. A goal can be an objective to reach (to reach a destination) or to be in a certain state (staying alive for fish). An agent is represented by the tuple $< W, G, A >$ where W is its representation of the world it is evolving into, G the set of all its goals g_i, and A the set of actions a_j it can perform. This tuple can be extended to include specific characteristics (time, conditional state, neighbouring functions...). An agent is characterised by a life-cycle composed of three steps: **perception**, **decision**, and **action**.

- **Perception**: This step represents the ability of an agent to collect information about its environment. Note that an agent's environment is a subset of the global environment, formally W. An agent has only a partial view of its environment. The agent's environment representation is updated using its perceptive function. Such a function can be as simple as a range around the agent, to complex functions representing real sensors. Perceptions also include communicated information from other agents. The first step for an agent is to update its knowledge base thanks to the perceptive function. A knowledge base represents information accessible and deduced by the agent. According to this, an agent decides its next actions. Considering the school of fishes example, a fish observes its environment to see if the distance between it and its neighbours are sufficient, and also to see if a predator is coming.

- **Decision**: During this phase, according to its knowledge and its perceptions, an agent chooses the actions it will realise in the action step. This can be deciding to go left or right, integrating an Artificial Neural Network, or computing dimensional solutions to a mathematical problem. Most algorithms focus on this step as it depends on the addressed problem and the characteristics required. In this step, an agent intends to get closer to its goal. Because an agent is autonomous, it has the choice to do nothing if it thinks it is better and so can refuse to do something. The decision process can be represented as rules (function linking W to G and A) or a more complex one. An example of a decision would be a CAV situated at an intersection and having to choose a path to reach a destination. Depending on local preferences, traffic state and passengers, a CAV chooses the path that is the most suited for it.

- **Action**: After its decision step, an agent executes the action it has decided to do. An action has an impact on the environment. This action effect can be anticipated whether the environment is deterministic, or

not (see section 3.5.2). In a deterministic environment, an agent can anticipate its action' effect. In a deterministic environment, it is possible to plan a set of actions with solid confidence in the result. From the CAV example, an action is to cross an intersection at a given time.

An agent can be **reactive** or **proactive** [161]. A **reactive** agent reacts to an external stimuli. Reacting to this, the agent follows its cycle until meeting a stop condition (waiting for another stimulus, achieving its goal,...). It reacts opportunistically to its perceptions of the environment. This is commonly known as reflex behaviour. Often, a reflex behaviour is light computational and requires small memory. A fish (from the school of fishes example) is a reactive agent. Indeed, it perceives a predator (receiving environmental stimuli) so it runs away from it. Artificial reactive agents follow a set of rules that are activated when receiving a stimulus.

Opposed to it, a **proactive** agent (also called cognitive agent) can proceed with complex reflections. It can modify, suppress or even create sub-goals. A proactive agent has its objective, the "meta goal" and it can create "sub-goals" that are some steps it follows to reach its "meta goal". Gleizes et al. [47] show an agent with the meta goal to make the maximum profit deciding to drop the sub-goal of buying an object to create the goal of buying another one. In the fleet of CAVs context, an example would be the CAV deciding to drop the goal of passing a vehicle to take the road leading to its destination. In this example, the meta-goal is to reach the final destination and the sub-goals are to pass a vehicle that prevents it to drive at a certain speed or to go left to save travel time. Because the exit is close, it has to drop one sub-goal.

The use of cognitive or reactive agents depends on the resolution means. The use of reactive agents implies to have lot of agents and the resolution comes from the interactions between them. MAS composed of cognitive agents often has a small amount of them as their tasks are more complex. A cognitive agent is aware of belonging to a social organisation, these agents are intentional; they have explicit goals and plans to achieve their goals.

The Environment

An environment can represent a multitude of scenarios, systems or problems. An environment can represent various world, from the real one like water for a school of fishes through the urban area for smart cities to mathematical function for an optimisation problem. The design of the environment is a primordial step as it models the addressed context. From an agent's point of view, the environment can be considered as "everything but me". The MAS as well as the agents have an environment. For the MAS, the environment is all the entities outside the system. While for an agent, it is the same as the system with the other agents added to it. Odell et al. [109] define an environment as follows: "An environment provides the conditions under which an entity (agent or object) exists". They specify two types of environment, the **physical** one and the **communication** one.

- The **physical environment** supplies laws and rules which govern and support (bear) the physical existence of the agents and the objects.
- The **communication** environment provides (1) the principles and processes that govern and support the exchange of ideas, knowledge and information, and (2) the functions and structures that are commonly employed to exchange communication, such as roles, groups, and the interaction protocols between roles and groups.

Weyns et al. [160] state that the *"environment is a first-class abstraction that provides the surrounding conditions for agents to exist, and that mediates both the interaction among agents and the access to resources"*. The environment structures the MAS and characteristics may come from the domain specification

or design choices. This stresses the fact that the environment is an independent building block in the MAS that encapsulates its clear-cut responsibilities, independent of the agents. Second, the environment provides the surrounding conditions for agents to exist. The environment is the part of the world with which the agents interact, in which the effects of the agents will be observed and evaluated. Three different forms of structuring are proposed by Weyns et al. [160]:

- The physical structure used to structure the space, topology and the possible distribution.
- The communicating structure refers to the infrastructure used to communicate. The network used, the infrastructure supporting stigmergy and the infrastructure for implicit communication.
- The social structure that refers to the organisational structure of the different entities.

As the environment has different characteristics that depend on the addressed problem, Russel and Norvig [128] have proposed four main properties to characterise the environment:

- **Accessibility**: this property is used to delimit agents' perceptions. An environment fully accessible signifies that an agent can perceive everything in it (for example an Intersection Management Unit can see the entire intersection). On the contrary, the environment is called inaccessible when agents have limited perceptions (a CAV relying on their sensors).
- **Determinism**: a deterministic environment means that the state of the environment is stably excepted from agents' actions (possibly known modifying rules). However, actions' consequences can be anticipated (like in a traffic light). Those characteristics are not ensured in a non-deterministic environment.
- **Dynamic**: a dynamic environment possesses a time characteristic with modification functions. It is subject to modifications through time without any convenience for the agents' process (i.e. a pedestrian crossing an intersection while a CAV processes if crossing is a good idea). A non-dynamic environment is called static.
- **Quantifiable**: an environment is quantifiable if it is possible to clearly define the number of perceptions and actions (an intelligent barrier that can be open or close and detects something or not). When the number becomes too large, the environment is said continuous (i.e. the totality of the ITS).

The environmental characteristics influence the agent design as they need to be able to take them into account. A strong cognitive agent may never finish its process before the dynamic environment is modified. In that case, a faster reasoning agent is to be preferred. Using this taxonomy, it is possible to select correct agents' characteristics.

Organisation

The organisation in a MAS is the context in which all collective behaviours are taken into account. Agents in a MAS are organised and from their organisation, a result comes. The objective of an organisation is to assist agents to reach their local goals and to control the agents while keeping their autonomy. The organisation can be **predefined** when the designer defines the role and the location of every agent in the MAS. In this case, the designer adopts an *organisation centred point of view* with a design at the organisation level. This means that the organisation, the patterns of coordination and the agent behaviours in the organisation are specified. But it also can be **emergent** when agents have the tools to self-organise. This is an *agent centred point of view*

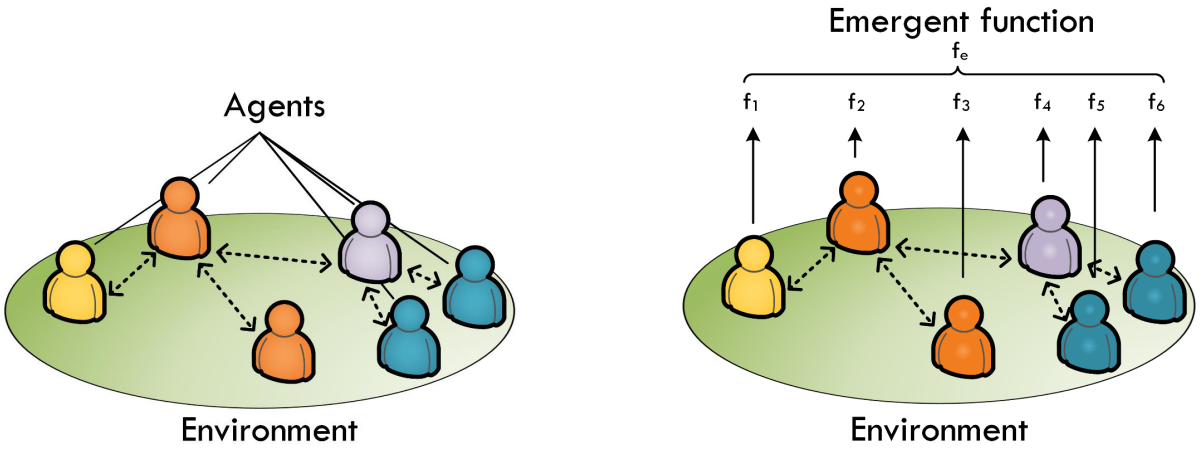


Figure 3.7: Multi-Agent System Scheme

design of the organisation. The system design is made at the agent level with a focus on the agent design and their interaction. In this case, the organisation is the result of the agents interactions. The figure 3.7 shows a MAS representation with an organisation of agents. Thanks to the organisation and the local function of agents, an emergent function f_e appears. If the organisation is different, the emergent function fe is likely to be different. For example a mathematical function, $f = 5 + 8 \times 7$ where agents are the operators. Depending on their location in the function, the result is either 61 ($f = 5 + 8 \times 7$) or 45 ($f = 5 \times 8 + 7$).

Di Marzo Serugendo et al. [135] define self-organisation as "*the process with which a system changes its structure without any* ***external control*** *to respond to changes in its operating conditions and its environment*". To be self-organised, a system must ensure three mandatory properties [133]:

- **Global organisation**: from the process of self-organisation the system converges to a stable state where it can fulfil its functional objective.
- **Dynamic Adaptation**: when confronted with a perturbation in the environment, self-organising system parts must be able to re-organise according to environment modifications to converge again to a stable state.
- **Lack of external control**: a self-organising system is autonomous and cannot be subject to the control of another entity or system.

Those three properties are mandatory but many other optional ones have been proposed (the book *Self-organising Software* [134] gives a strong understanding of the subject). From those optional properties the *decentralised control*, *simple behaviour* as well as *local interaction*, *adaptability* and *feedback loops* are relevant for this thesis.

- **Decentralised control**: this is a primordial characteristic as a global control often leads to a bottleneck in the case of a high number of agents.
- **Simple behaviour and local interactions**: to be as close as possible to reality, the system's parts only have a local view of the environment (opposed to a global view). As the system functionality emerges from agents' interaction, simple behaviour is often preferred.

- **Adaptability**: close to the dynamic adaptation property, this one enables that a particular organisation is maintained despite environmental changes. This can be close-up to resilience as the adaptability of a system improves its resiliency.
- **Feedback loops**: in a non-deterministic environment, the effect of an action cannot be anticipated with confidence. Thus components (entities and agents) need feedback from the environment to understand the consequences.

As stated, in a MAS the solution comes from the interactions of agents evolving in it. Depending on the environment nature, interactions may take different forms. Agents use the environment as a mean to interact according to the communication structure. As an artificial system, several assumptions about the environment are made because they are not in the scope of the addressed problem. Thanks to these assumptions, it is possible to focus on the problem instead of the structure.

Common Assumptions and Limits about Agents Interactions

A common assumption concerns the interaction between agents. Usually, in a MAS, communication between agents is reliable thus ensuring **understanding**, **absence of errors**, and **no delays**. Within a simulation of a real environment, communication is designed from scratch and simulated, so it can be considered as perfect. However, in real systems, agents may need to use a network with every issue coming with it (errors, delays, congestion, etc). The assumption is made on the basis that this problem is not included in the system functionality but is delegated to another one. Network issues have been known and addressed for a long time to reduce the network limits.

The assumption of no malevolence signifies that no agent has the purpose to disrupt the system. The consequence is that agents trust other agents and do not lie. Even if, in the particular case of competitive MAS, this assumption is no longer valid. Assuming that other agents are supposed to be trustworthy, it enables to implement cooperation and negotiation between agents. Several strategies have been designed to identify and exclude malevolent agents from the cooperation scheme. An example of a malevolent agent in the context of ITS could be a CAV lying about a priority to prioritise itself and making another agent wait for nothing.

Those assumptions, understandable communications and not malevolent agents, highlight some limits when designing MAS. As stated before, the MAS functionality emerges from the interactions of its components. If these interactions are altered, the emergent phenomenon would not appear.

3.5.3 Adaptive Multi-Agent System

This section presents a particular type of the Multi-Agent System known as Adaptive MAS (AMAS). The AMAS approach is an organisational approach to build complex systems that adapt, continuously and locally, to the dynamics of their environment. It focuses on the interactions between the system and its environment on one hand and between the parts (agents) of the system on the other. These interactions are based on local and cooperative information processing by the parts of the system, which only have a partial view of their environment. This principle of locality guarantees the emergent nature of the system's operation [1].

This extension has been derived with the purpose to look at MAS which self-adapts to perturbations with properties of *decentralised control*, *simple behaviour and local interaction*, as well as *adaptability* and *feedback loops* [45]. AMAS is used to tackle problems requiring **scalability** and **adaptability** (including **openness**). With fidelity to the bottom-up design described by the MAS approach, AMAS has been developed to explore agents' cooperation more deeply using the concept of *criticality*.

In a MAS, agents interact continuously with each other through the environment because they share it and an interaction also may come from an agent's desire. An agent has only a local view of its environment, its

goal and, limited capacities due to the distribution of capacities among all the agents in the system. To reach its goal, an agent has to choose actions that bring it closer to its goal. However, many causes can prevent an agent to execute the desired action:

- 1) the agent cannot execute the desired action and needs the help of another agent (lifting an object too heavy for one agent);
- 2) the presence of another agent prevents the agent to execute the action (a CAV blocking the way);
- 3) the goal of another agent is incompatible (crossing an intersection at the same time);
- 4) another agent is in competition (being the first to reach a destination), this occurs only in competitive MAS;

Excepted for the fourth one, agents need to **cooperate** together to help everyone for reaching their goal. In the AMAS approach, the cooperation between agents is a crucial part of the design.

Cooperation in AMAS

Cooperation can be seen at two levels. At the MAS level, cooperation is related to the MAS' influence on the environment (precisely on the system's functionality). The MAS is considered cooperative with the environment if its effect is at least beneficial to the functionality. Refining the granularity, the MAS emergent function needs to be cooperative meaning that agents' actions must be cooperative. Concretely in the decision step, a cooperative agent cannot choose an action that it does not think it is cooperative. Actions can be categorised into three classes depending on their effect [71]:

- **Cooperative**: an action is considered cooperative when it has a positive effect on other entities affected by it.
- **Neutral**: with a neutral action, an agent does not modify other entities' functionality.
- **Antinomic**: an antinomic action, as opposed to a cooperative, affects negatively another entity and its activity.

However, because of the agent local perceptions, an agent may realise a cooperative action that has a non-cooperative effect as a result. This is especially true since in a non-deterministic environment, the decided action may have a non-cooperative effect not anticipated by the agent. There is a clear distinction between the decision and the actual effect on the environment. Using the feedback loop, a cooperative agent tries to make it right and may avoid choosing the non-cooperative twice. This is not always true when talking about a coalition of agents [71]. In a coalition where agents make a joint decision, an agent may realise an antinomic action because it has a personal gain and the overall coalition effect is still cooperative (the negative effect induced by the antinomic action is compensated by the coalition). A system is in a cooperative state only if all actions between the parts are cooperative .

The AMAS theory has been studied and enriched through the years is based on the **functional adequacy** of a system. A system functionally adequate is efficient by nature to complete its tasks. The functional adequacy of the system refers to the system capacity to realise the function it has been designed for. Glize [48] gives a theorem of functional adequacy with a cooperative vision: "*A system is functional adequate if it has no antinomic activities in its environment*". This means that to be considered as a functional adequate system, not only the system has to execute its function but it has to never cause additional problems. As he stated,

the minimal property for a system is to not disturb its environment. Then he adds that *"to any functionally adequate system in a given environment can be associated with at least one system having a cooperative internal medium and achieving an equivalent function"*. In consequence, it is possible to design a system by focusing on cooperation in its internal state and ensuring its functional adequacy. This provides the possibility to use a bottom-up design instead of the usual top-down, ensuring cooperation in the internal state of the system. In the MAS context, the bottom-up design focuses on the agent design and ensuring that they are cooperative to create a cooperative state. The AMAS approach gives a method to design agents cooperatively, thus ensuring the functional adequacy of the overall system.

Ensuring that the internal part of the system is cooperative requires that agents only act in a cooperative way, which can be difficult to achieve depending on the environment. Agents may be stuck in a situation with no other choices but to act non-cooperatively (the functional adequacy is no longer ensured). For example, if a CAV blocks a one-way road with two other agents that want to pass oppositely, whatever the CAV does another CAV activity is prevented while it remains on the road. These situations are called as Non-Cooperative Situation (NCS) and are summarised in the table 3.8. First iterations have been proposed by Glize [48] and enriched by Mefteh [96].

As an agent must be cooperative, if it is in a cooperative state it performs the desired action but if it is in a non-cooperative state it tries to return to a cooperative state. From the functional adequacy theorem, the presence of a NCS leads the system to be in a non-cooperative state: the system is not functionally adequate. The AMAS approach provides several tools to enable agents to overcome NCS. The concept of *criticality* is used by agents to solve encountered NCS. The *criticality* is the estimation of the satisfaction of an agent about its objective. Every agent can computes a *criticality* value using a function to estimate the distance with its objective. This value can be shared with neighbour agents. With all agents exchanging their criticalities, they can cooperate to solve NCS. Usually, an agent tries to help the worst critical agent in its neighbourhood, and ensures that the action it decides does not create a new worst critical. To be efficient, it is mandatory that agents normalise their *criticality* value, thus allowing to compare a *criticality* with another.

A methodology called ADELFE [116] has been proposed to help and guide the designer in the AMAS design. ADELFE is a French acronym that means "Toolkit for Designing Software with Emergent Functionalities" ("Atelier de DEveloppement de Logiciels à Fonctionnalité Emergente" in French). Initially, ADELFE has been inspired by the *Rational Unified Process* and included 5 main phases with twenty-one activities and producing twenty-six work products. It has been extended by Bonjean et al. [19] and the last version, Adelfe 3.0 by Mefteh et al. [97], added a sixth phase and a possibility to return to a phase with a condition. These six-phases are the following:

- **WD1 - Preliminary Requirements**: Clients, users, and designers collaborate to clarify the system requirements (function, design, limits, and constraints).
- **WD2 - Final Requirements**: The Business Analyst gives a detailed description of the system environment. This phase concludes on the efficiency of the MAS approach to address the problem. This is achieved by a MAS specialist.
- **WD3 - Analysis Phase**: This step analyses and concludes on the AMAS adequacy to solve the problem. In this step, agents are identified and defined among previously defined entities. Looking at the local level, an AMAS specialist states the relevance of using the AMAS approach.
- **WD4 - Design Phase**: A detailed architecture is produced at this phase including communication acts, a module view, and different behaviours.
- **WD5 - Implementation Phase**: This phase aims at producing the system according to the different

Non Cooperative Situation	Definition
Perception	
Incomprehension	The information structure is unknown for the agent.
Ambiguity, Uncertainty	The agent doubt on how to interpret the received information.
Ambiguity, Ambivalence	The axiome received leads to multiple possible conclusions.
Decision	
Unproductiveness	No new knowledge is extracted from an interpretation. Either the information is already known, uninteresting or not usable as it is (another information is required).
Incoherence	An interpretation is in contradiction with the agent's knowledge base.
Incompetence	The agent has no available actions with its current knowledge base.
Internal Conflict	From its knowledge, the agent decides several activities that are not compatible between them.
Action	
External Conflict	The action is not compatible with one of its neighbours
Concurrency	The same action is performed by two agents simultaneously
Uselessness	The action has no effect for the agent or its neighbours
Inability	The agent has decided an action that its set of skills not include

Figure 3.8: Classification of Non-Cooperative Situations

characteristics specified in previous steps. First, the AMAS framework is constructed and then agents' behaviour is implemented.

- **WD6 - Living-Design Deployment**: The concrete deployment in a real situation is achieved. This last phase is application-dependent and may require the intervention of an expert (and the client).

Following the ADELFE methodology, one can identify why and how an AMAS provides an advantage and how to design it. As only an overview of the different phases has been given here, more information about activities and work are available in [19] and [97].

Related AMAS Works

Many AMAS have been designed and used in various contexts. Even if AMAS is not currently theoretically proven (to prove it requires a powerful tool not yet available) the success of the application for many different problems type in various contexts, tends to validate the approach. Currently, the validation only comes from experimentation. Among the recent AMAS already designed can be cited:

- Adaptive Multi-Agent Systems for Wind Power Forecasting [35].
- Self-structuring of multi-objective and multi-criteria real-time traffic in a virtual world [31].
- Investigations of Process Mining Methods to discover Process Models on a Large Public Administration Software [105].
- Lifelong Learning by Endogenous Feedback Application to a Robotic System [29].
- Multi-Criteria and Multi-Objective Dynamic Planning by Self-Adaptive Multi-Agent System, Application to Earth Observation Satellite Constellations [20].
- Self-organisation of robotic devices through demonstrations [153].
- Adaptive Multi-Agent System for Hybrid IoT [53].
- An adaptive multi-agent system for self-organising continuous optimisation [70].

3.5.4 Conclusion on the AMAS Approach

The AMAS approach has been studied and successfully used for many years to address complex system problems. AMAS are particularly relevant for solving problems where many autonomous entities are involved. The use of a bottom-up design focusing on local interaction between entities called agents allows addressing characteristics of **dynamicity, openness, privacy and scalability**. Indeed, this is justified by;

- **Dynamicity:** To be functionally adequate, the system must be in a cooperative state. Dynamic changes may disrupt the system and leads it into a non-cooperative state. An agent does its best to return to a cooperative state, so the non-cooperative situations are solved when a new cooperative state is reached.
- **Openness:** With a focus on the agent design, the arrival or the departure of an agent does not disrupt the internal functioning of an agent. Instead, the environment is modified and the agent needs to adapt to it.
- **Privacy:** An agent is autonomous with a local behaviour. It does not rely on another entity to decide, which means that it does not have to exchange an information if it does not want to.

- **Scalability:** Scalability is ensured thanks to 1) the distribution of computation among agents, preventing the appearance of a bottleneck of decision; and 2) the simple behaviour of each agent used to create multiple interactions that solve the problem.

The CAVs characteristics are similar to the one of an agent in AMAS. A CAV is autonomous thanks to an Autonomous Driving Process (ADP). Its goal as well as its characteristics are private to cope with the protecting data laws. CAVs are moving entities thus leading to **dynamicity** and openness **constraints**. By focusing on local interactions and agents' behaviour it is possible to avoid the use of a central entity managing everything. This is primordial in ITS context as the number of entities can grow in a way that communication and computation time gets so heavy that it is no longer suitable. In this thesis, a CAV is seen as an agent which interacts locally with other CAVs using V2V communication. This allows one to stay close to reality with CAVs without centralising the control. In the following sections, the proposed architecture taking into account the two AMAS-based modules answering both challenges of **cross-understanding** and **communication optimisation** is described.

Chapter 4

Communication Transformation and Optimisation in a Fleet of CAV by Adaptive Multi-Agent System

This chapter presents two AMAS that is the contribution of this thesis. The overall architecture with these two AMAS-based modules is presented to understand how they can be added to an existing CAV and each module is then detailed. The first module, called Learning Usefulness of DAta (LUDA) has been designed to address the challenge of **cross-understanding**. The second, called Cooperative DCOP (CODCOP) establishes a cooperative behaviour for the fleet of CAVs to **optimise communication**.

4.1 A Social Fleet of CAVs

This section presents the two main contributions of this thesis: a first module based on AMAS called LUDA (Learning Usefulness of DAta) for **cross-understanding** and a second AMAS called CODCOP (Cooperative DCOP) for **communication optimisation**. A CAV is composed of an Autonomous Driving Process (ADP) responsible for all decisions and to which both modules are linked. First, a formalisation of the environment and the agents is given to provide a concrete understanding of challenges, constraints, and possibilities. From this formalisation, an architecture composed of the two proposed modules is detailed. The agentification of CAVs into agents as well as the architecture are detailed before the description of the modules. The first module, called LUDA, focuses on the transformation of the referential frame to enable an agent to understand other CAVs. The second module has to optimise communication so that only crucial information is shared, meaning without congesting another CAV with too much information. This section first presents the *agentification* of CAVs into agents.

4.1.1 Agentification of a Social Fleet of CAVs

Intelligent Transport Systems (ITS) and more particularly fleet of CAVs are complex systems involving more and more intelligent entities. CAVs are equipped with an increasing number of sensors making them sentient, able to gather an amount of information that a human is not able to perceive.

The objective here is to address cooperation between CAVs that use V2V communication. Many works are based on communication and cooperation between CAVs to improve the overall system functionality [14] [121] [103]. Sharing useful information results in an improvement in some traffic characteristics like improving the traffic flow or reduce the pollution. CAVs rely on shared information to improve their behaviour. In the addressed system, a CAV is an agent selecting which information to share with others (related to the challenge of **communication optimisation**). It perceives environmental information in addition to its state (all information about it like its speed, energy remaining, etc). The CAV agent is also responsible for the value of received information (related to the challenge of **cross-understanding**). It can transform the value to be usable before this one is given to the ADP.

A perception can represent any environmental information, the semantic of information is not addressed in this work. Indeed, a CAV agent does not need to know the structure of an information but only its value. Indeed, the semantic is only used by the ADP to compute a decision. For example, a CAV agent CAV_A receives an information i with a value v and a label l. It has to transform the value v and give this information with a value v', modified according to the referential frame, to the ADP with the label l not modified. In this case, the label is only used by the ADP. CAV_A can also perceives an information i with a value v and a label l. It has to decide to share it or not. The CAV_A shares the information i if it thinks it is useful to an ADP. In consequence, CAV_A does not need to know the information structure but only if it is useful, this knowledge is provided by the ADP. In consequence, for both actions of **transforming a value** (cross-understanding objective) and **deciding to share a perception** (communication optimisation objective), the CAV agent does not need to know the information structure, in consequence, this point is not addressed in this thesis.

A CAV agent is defined with the following characteristics:

- A set of perceptions $P_1 = \{p_{1,1}, ..., p_{k,1}\}$ of the agent (in this case, the agent 1). This set of perceptions is derived from the set of information available in the environment. Because perceptions are locally made by an agent, the value of the information perceived may be different from the environment one because of errors/noise.
- A function F representing the referential frame of the vehicle. This function is applied by the agent to the environment's set of perceptions to obtain the agent's set of perceptions. This function is not known by the agent.
- The range r of communications V2V of the CAV.
- A set of neighbours $N = \{n_1, ..., n_l\}$ of the agent. The neighbours of an agent are the agents within its range of communication.
- The set of communicated information from other CAVs in the fleet, $Pc = \{pc_{1,1}, pc_{1_2}, ..., pc_{k,n}\}$, with $pc_{k,n}$ the information k perceived by the neighbour n.
- The set of actions $W = \{w_1, ..., w_m\}$ available for the agent. All these actions are communication-related. As an entity responsible for communication, the agent chooses how to interact with other agents. It can communicate a subset of its perceptions to other agents. Or transform the value of a pc before giving it to the ADP.

In addition to these characteristics, a CAV agent is possibly limited by external constraints, such as network ones. A CAV is a critical system (it may have only a short time before taking a critical decision) with limited memory and computing capacity to make a decision and do another task. Moreover, with the constantly increasing number of devices embedded in the vehicle, the energy demand becomes a limitation that also needs to be addressed. It cannot be heavy computational and cannot ask for a great number of

resources to have enough time to process a decision. Furthermore, only a limited amount of information can be processed while deciding as the time is limited. This induces the problem of selecting only a small amount of information to communicate. If the total amount of communication exceeds the capability of the receiving agent it may not have the time to process an important information.

4.1.2 The Environment

The environment is constituted of roads allowing vehicles to move on it and other road users. A road possesses many characteristics including the material used to build it, the state, weather conditions, size, lane, infrastructures, signalisation, and many others. Naturally, entities evolving in this environment include vehicles of all sorts (cars, buses, cycle,...). Note that some vehicles are driven by a human driver and others are CAVs. According to Russel and Norvig [128], the environment considered is:

- **Inaccessible**, CAVs have only a partial view and knowledge about their environment. Depending on the entity, perceptions come from their embedded sensors or from its neighbours communication. Sensors are subject to failure, misconception and limits that need to be represented in the system. For example, the detection of another CAV's position may be wrong.
- **Indeterministic**, agents' actions are assumed to be indeterministic because the control cannot be assumed as perfect, making the environment **indeterministic**. This assumption is made on the basis that the control and the result are depending on the ADP outside the addressed system. In consequence, it is nearly impossible to know in advance the result of an action.
- **Dynamic**, the environment is continuously evolving with or without agents' actions. Entities can leave or enter the system making it **dynamic** because subject to environmental perturbations. The dynamic is also due to the weather modification making the luminosity evolves through time for example. Moreover, all the non-CAV actors that modify the environment through their actions make it dynamic.
- **Not quantifiable**, even if technically the number of information available is finite, the number is large enough to be considered as not finite. Moreover, it is impossible to predict all future CAVs advancements, making it not possible currently to know the precise amount of required information.

Now that the overall environment's characteristics have been stated a formalisation is provided. This formalisation aims at giving an accurate and understandable representation of the environment far from the CAV specificities. A fleet of CAVs can be seen as a coalition of agents linked together using V2V communications. All CAVs in a fleet do not share a common objective but cooperate. The environment includes:

- Let $P = \{p_1, p_2, ..., p_k\}$ be the set of all perceptible information in the environment. An information is represented by a label and a value (i.e. speed : 50 km/h).
- The period during which information about the environment changes (i.e. the weather). That can be within a second, a minute or a hour. The set $T = \{t_1, t_2, ..., t_{max}\}$ is the set with all the time where the environment evolves alone. For example, at the time t_3, it begins to rain. The set T is not known in advance and not accessible by the agents in the environment.
- $A = \{a_1, a_2, ..., a_n\}$ the set of agents.

Agents and the environment being formalised, the next step is to study how each CAV agent can be equipped with capacities to understand any communication and to select the most useful one to be communicated. Hereafter is presented the CAV agent architecture with the two proposed modules.

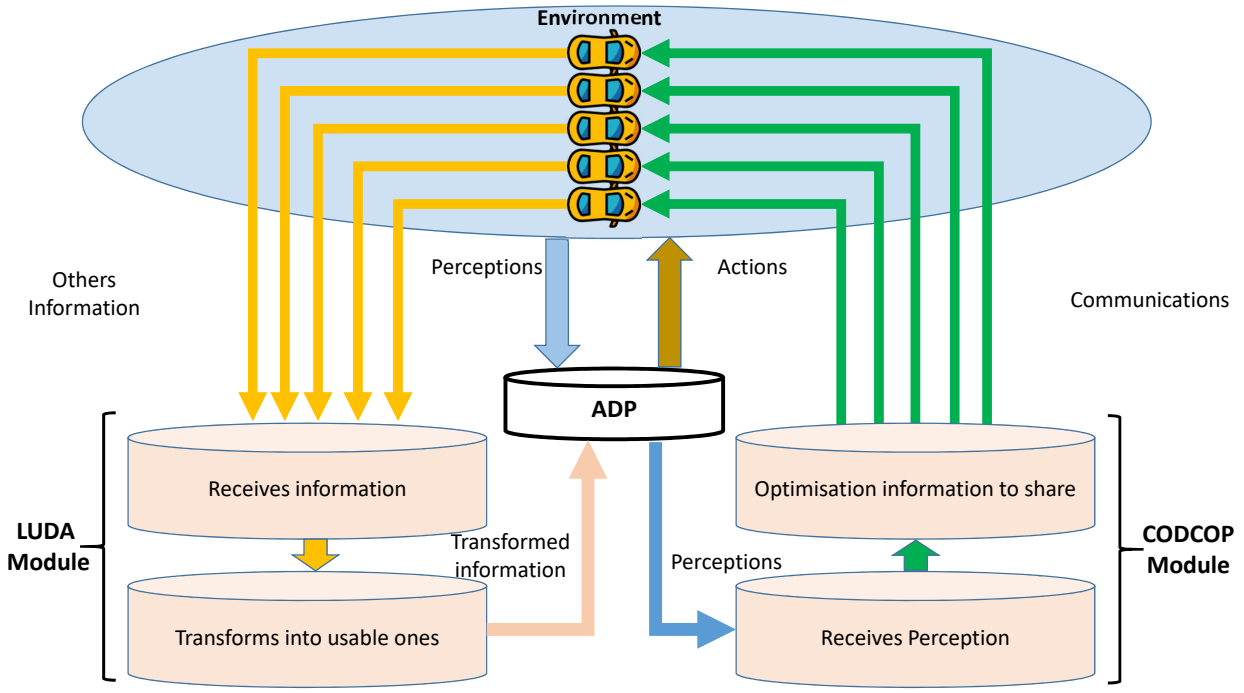


Figure 4.1: Global architecture of a CAV agent with the LUDA and CODCOP modules.

4.1.3 CAV Agent Architecture

The CAV architecture composed of three main parts can be seen in the figure 4.1. The middle one, in black, is the Autonomous Driving Process (ADP), responsible for vehicle autonomous driving. As we have already seen, this part is out of the scope of this thesis. The first module named LUDA has for objective to learn how to transform information from other CAVs into a useful one. This means that the transformed information can be sent to the ADP to improve the agent behaviour. This module addresses the **cross-understanding** challenge. The second module, CODCOP, receives perceptions from the driving process and is connected to other CAVs in the environment. The CODCOP module selects information that is the most useful without overloading another agent.

4.2 A Self-Adaptive MAS for Cross-Understanding

This section presents the LUDA module that aims at transforming an information communicated by another agent into an information that can be used by the ADP and thus, improving the decision [3]. Consider one agent CAV evolving in its environment. According to the AMAS approach, its neighbours, other CAVs, try to be cooperative. That means they send information they perceived and they believe useful for sharing with others. By increasing one's understanding of the world, the overall system aims at being more cooperative thus more efficient.

To illustrate it, imagine the situation where a pedestrian is crossing the street. The crossed street is hidden from the CAV, called CAV *A*, coming from another road and arriving to the intersection with the street, so *A* has no information about the street. Cooperation can be realised if CAV *B*, having the pedestrian information, can use V2V communication to inform the CAV *A*. Thanks to this information, the ADP of *A* can anticipate the situation (the pedestrian crossing the street) instead of reacting to it in emergency. Because it knows it must stop in soon after to let the pedestrian cross, *A* can give the priority on the intersection to another CAV which would have been stopped otherwise due to the usual signalisation. By only following the signalisation the CAV *A* action would have been a non-cooperative action because of disturbing the environment. Thanks to V2V communication the action taken has been more cooperative.

The example illustrates the necessity for local cooperation to take place in a fleet of CAV. A statement has been made on the problem of sharing information and to use it in the absence of a shared referential frame. From the same example, CAV *B* needs to give additional information about the pedestrian to be accurately understood used by others. However, sending this has a cost (especially in the context of broadcast in ITS and smart city) and a CAV cannot share all the perceived information. Following the idea of edge computing, the transformation is realised by the receiving instead of the sender, to avoid a bottleneck of communication [1].

The LUDA module objective is to enable an agent within a MAS to translate an information from another agent's referential frame into its own. To evaluate if the objective is achieved, according to the AMAS approach, no agent in the system must be in a non-cooperative state. Moreover, because an agent can translate several referential frames, the same information can be redundant. In a noisy environment, redundancy can be used to lower the error level. To cope with the fleet of CAVs context, the challenges that need to be addressed in addition to transform information are: **dynamic**, **openness** and **scalability**. This section proposes a formalisation of the problem; then it presents the architecture defined to solve it as well as the behaviours of its component; from this, the idea of the emergent resolution coming from each entity behaviour is given. Note that experiments used to validate the proposed solution will be presented in chapter 5.

4.2.1 From Intelligent Transport System to Multi-Agent System for Cross-Understanding

The idea behind the formalisation is to transform a transport-related problem into a mathematical more suited one to be addressed by data estimation methods. The ADP of a CAV is considered as a black box which has the responsibility of computing a decision. Processing a decision is a complex task involving a wide variety of information aggregated to obtain a result for each vehicle's effector. More accurate information is available, better results would be and thus driving. The LUDA module aims at being integrated between the reception layer (gathering other vehicles' communication) and the ADP. By transforming the communicated information it provides to the ADP information in a way that can be usable. So, the LUDA module gives access to more information to the ADP than it would have without it.

The set of perceptions P of a CAV agent is composed of two subsets, here called P_D and E_D, such as $P = P_D \cup E_D$. The first subset called **Principal Data** ($P_D = \{p_{d_s}\}$) refers to information directly linked to the agent sensors (s being one of the agent sensors) and which this agent can understand. The second subset called **Extero Data** ($E_D = \{e_{d_j}\}$) refers to information communicated by its neighbouring agents (j being

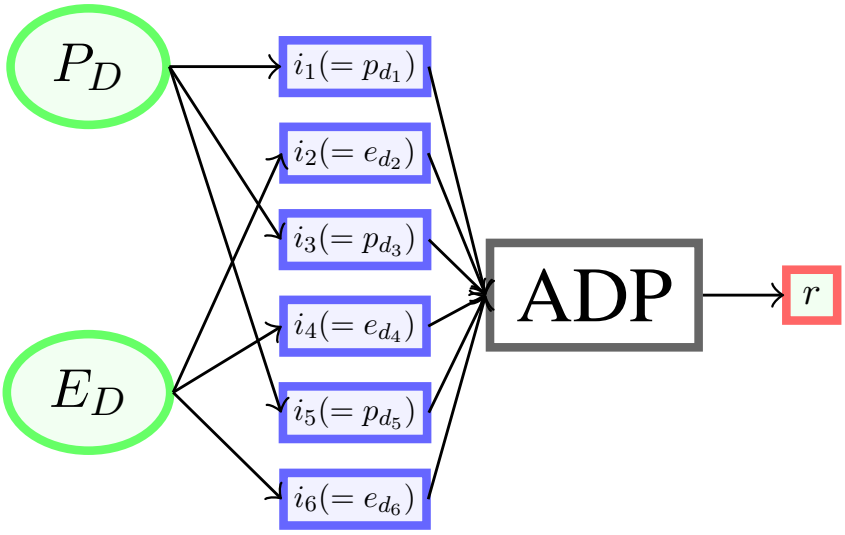


Figure 4.2: The ADP computes the result r using principal data for i_1, i_3, i_5 and extero data for i_2, i_4, i_6.

one of the neighbour agents), each having a different referential frame. The cross-understanding problem corresponds to an agent being able to select from E_D missing information of P_D while translating them into its own referential frame.

The main objective of cross-understanding is to enable an agent to use e_{d_j} communicated by a neighbour agent j to replace a missing p_{d_s} for its ADP. The ADP uses a function (unknown to the LUDA module) and computes a result r to solve a situation. The ADP takes a fixed number of information $I = \{i_1, ..., il\}$ as input. The value of an input i can come from P_D or E_D; for example i_1 can takes its value from p_1 or e_1. The figure 4.2 illustrates an example with 6 inputs, 3 p_{d_s} $(p_{d_1}, p_{d_3}, p_{d_5})$ and 3 e_{d_j} $(e_{d_2}, e_{d_4}, e_{d_6})$.

When all principal data are available the computed result is considered as the ideal one noted r_{ideal}. When some of principal data are missing, the objective is to find the right extero data e_{d_j} to replace them while minimising the difference between r_{ideal} and r (obtained by the ADP using e_{d_j}). For an agent, the problem can be formalised as follow.

Given $P_D = \{p_{d_s}\}$ and $E_D = \{e_{d_j}\}$

For each $p_{d_s} \in \{P_D\}$**:**

1) select $e_{d_j} \in \{E_D\}$ that can be linked to p_{d_s}

2) determine the two coefficients x and y such as $p_{d_s} = x \times e_{d_j} + y$

While minimising :

$$error = \sum_{k=0}^{nbInput} |p_{d_s} - e_{d_k}| \tag{4.1}$$

As different and several e_{d_j} can be available for one p_{d_s} the LUDA's second objective is to take advantage of this multiple sources to improve the accuracy of the proposed values for p_{d_s}, especially in a noisy environment.

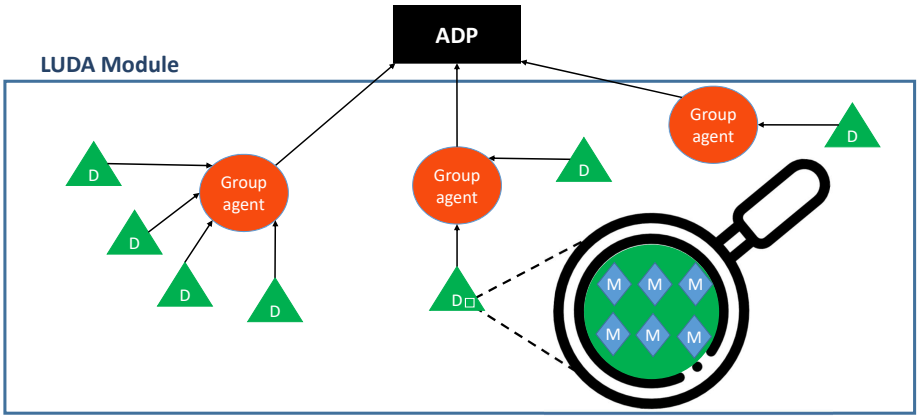


Figure 4.3: Example of LUDA module architecture. The ADP is represented with a black rectangle. Connected to it are *groups agents* that give a value for an input of the ADP. The computed value comes from the combination of all grouped *data agents* values transformed for common input. This value is transformed thanks to the *morph agents'* behaviour, adapting a function to predict the objective value as close as possible.

4.2.2 The LUDA Architecture

The LUDA module is an Adaptive Multi-Agent System composed of three types of agents:

- *Data agent*: it represents an information perceived or communicated, its role is to find other *data agents* that represent the same environmental information, in that case they are *related*. For example, $e_{d_4} \in E_D$ and $p_{d_1} \in P_D$ represent the same environmental information with two different referential frames. The *data agent* da_A representing p_{d_1} needs to find the *data agent* da_B representing e_{d_4} because they both represent the same environmental information. A *data agent* is **inactive** when its information is missing.

- *Morph Agent*. It is created by a *data agent* and its role is to transform the value of its *data agent* creator into the value of another *data agent*. For example, it uses a function to transform e_{d_4} into p_{d_1}.

- *Group Agent*, its role is to group together all the *data agents* that represents the same environmental information and to send a value to the corresponding input of the ADP. For example, a *group agent* uses the values given by e_{d_4} and p_{d_1} to give a value to i_1.

An example of architecture is given in figure 4.3. The three types of agents cooperate to achieve the LUDA module's objectives. Each agent is detailed in the next sections. The system's environment is a numerical representation of the communication and the autonomous driving process system as seen in figure 4.2. Note that every agent has only a partial view of its environment. For example, *group agents* are the only ones to perceive the ADP but they do not see morph agents.

The global objective of the system is to find an organisation where *related data agents* are grouped together. Figure 4.4 illustrates the mechanism for the system to obtain an organisation able to provide an accurate estimation in case of missing data in the environment. From this organisation, it is possible, when an information is missing to use the value from other *grouped data agents* for an input i. The value from a *data agent* having an information from E_D needs to be transformed using a *morph agent's* function. To detail figure 4.4, in step ❶ *data agents* perceive if their related information is available in the environment (the corresponding p_{d_s} or e_{d_j}). This is the direct consequence of a perception realised by the *CAV agent*. If the CAV agent has perceived an information, the related *data agent* is considered active and it activates all

Figure 4.4: Overview on the global organisation process used in the LUDA module.

its *morph agents* that behave in step ❷. In this step, each *morph agent* makes a prediction of the value of another *data agent* using its transformation function. After, a feedback is given to the *morph agent* about the true value, then it adapts its function to improve it. Once they have finished, step ❸ begins and the *data agents* evaluate if the group they belong to is convenient for them. If not, they look for another group, more suitable. Finally, in step ❹ data agents re-organise themselves among group agents. The different agents will be described in detail hereafter.

4.2.3 Data Agent

A *data agent* represents an information (from P_D or E_D) that is either perceived or gathered from communication. Formally a *data agent* corresponds to one p_s or one e_j. It can be activated if the information it represents is available in the environment (the CAV agent perceives it). When it is not, the agent is inactive as it has no value. The local objective of a *data agent* is to be in a group that it considers *homogeneous*. A homogeneous group means that all the *data agents* inside are representing the same environmental information. This means that a *data agent* has to find a way to change its value according to other related *data agents* in its environment. Note that here two *related data agents* means that the information they represent is an interpretation of the same environmental information but with a value according to their referential frame (for example, 50km/h becoming 31.07 mph). For example, a *Data agent A* observes other active *data agents* in its environment (respectively *B* and *C*) and perceives their value. With this information, the agent *A* creates a *morph agent* that will help the *data agent A* to transform its value into the one of *B*. The process is the same between C and B, as well as B and A.

The creation of a different type of agent is motivated to cope with scalability. With a high number of *data agents* in the environment, each *data agent* would have to do the adaption process for each of them. This can lead the agent to use too many resources and time, and to prevent it to provide a value if required. To address this issue is to add another type of agent, called *morph agent*, one for every neighbour *data agent* in the environment. This add is driven by each *data agent* individually that creates a *morph agent* linked to it and another *data agent* in the environment. This creation is realised each time a new *data agent* is perceived. For example, when agent *A* perceives for the first time the agent *B*, it creates the associated *morph agent*. When two *data agents* are confident about each other (they think they represent the same environmental information) they form a group. They are confident about each other if they can accurately predict the other one value, using their *morph agents*. For example, if agent *A* and agent *B* can predict the other one value with low errors, they will group. The group is managed by a *group agent*. A *data agent* has only a partial view of the environment. It includes the *group agent* it is linked to, neighbour *data agents*, its related information, and its *morph agents* it has created.

4.2.4 Morph Agent

A *morph agent* is created by a *data agent* to transform its value into the value of another *data agent*. When a *morph agent* is created, it is associated with two *data agents*; d_c its creator and d_o its objective one, a neighbour of its creator. A *morph agent* is activated by its creator when the *data agent* objective is also activated. The local objective is to find a transformation function that makes it able to obtain the value of d_o from the d_c's value. To do this, a *morph agent* computes a linear function that approximates the value according to the precedent values of d_o. With the linear function 4.2, a *morph agent* can make a **prediction** of the d_o's value from the d_c's value. The form of the function is presented in equation 4.2 with x and y the linear coefficients. Note that this equation does not include the μ coefficient used to model the likely noise affecting the d_o. This lack is explained because it is not the role of a *morph agent* to address noise but the *group agent* that uses several *data agents* values. The local objective of a *morph agent* is to make the closest prediction for its d_o. By increasing its accuracy, it increases the confidence of its *data agent* d_c for the d_o. Thus, making it willing to group with the *data agent* d_o. In consequence, a *morph agent*'s result has a direct impact on the *data agent* d_c behaviour. Moreover, because the number of *morph agents* increases exponentially, using the most simple computation helps to limit the resources and computation impact.

$$d_c = d_o \times x + y \tag{4.2}$$

A *morph agent* has the capacity to store a limited amount of examples in its history. The history of a *morph agent* is a collection of examples. An example is a triplets: d_o value; d_c value; $prediction$ the result of the function with d_c as input. When a *morph agent* has access to all this information, in other terms, when d_o and d_c have their related information value available, it creates this triplet and stores it in its history. Using this example, a *morph agent* is able to modify its coefficients x and y.

A *morph agent* has a local objective, achieving the smallest difference with the d_o value. In consequence, it has the possibility of evaluating how far it is from it. This is known as the agent's *criticality* in the AMAS approach. Criticality is used to enable cooperative behaviour by giving the idea about an agent's state. A *morph agent* computes its criticality according to equation 4.3. It corresponds to the maximum distance between all its predictions in its history and the real d_o value observed for the example. The criticality is normalised using the range of values of d_o. This range corresponds to the distance between the minimum and maximum value of all d_o in the *morph agent*'s history. Using this normalisation, the criticality value is likely to be between 0 and 1 (0 is perfect and 1 is a bad prediction) and it enables the comparison of criticalities between two *morph agents* even if the values are highly differents (if greater than 1 then the prediction is really bad). For example, the *morph agent M* transformed the value 10 of its d_c into the value 15. Then it compares it with the value from its d_o which is 20. From its history, the *morph agent* evaluate the range at 50. In consequence, this prediction criticality is evaluated at 0.1 ($|15 - 20|/50$).

$$crit(ma) = argmax(\frac{|(x \times d_c + y) - d_o|}{range(d_o)}) \tag{4.3}$$

The process used to adapt coefficient is depicted in the algorithm 1. The first thing to notice in the algorithm is the use of only 3 examples to modify the coefficients. The number 3 is the minimum to do a linear adaptation. As stated before, there is a need to limit the time and resources used to perform the LUDA module behaviour. Thus, using 3 examples only ensures that the increase of examples number do not impact the computation time. To select which example from the history to use, a *morph agent* seeks for the three examples that its function gives the worst prediction for. This is accurate with its objective of minimising the maximum difference of all the examples in its history. Thus, by adapting the transformation function to improve the three worst examples in its history, it is likely to lower the criticality. The first step is to use a linear regression method to compute the *best* coefficients value for those three examples (line 2 -

13). However, replacing the old coefficients with these new ones is not suitable. Because always replacing the old coefficients with new ones will only result in the transformation function alternating between two models, old coefficients are only modified (lines 14 and 15). Old coefficients are modified according to the new coefficients and a coefficient ω. ω takes a value in the range $[0.1, 0.9]$. The higher the ω's value the higher the impact of new coefficients. Then a new prediction, using new coefficients x and y, is made for all examples in its history (line 16) allowing to sort them in descending order.

Algorithm 1: Morph Agent Adaptation

Require: $nbexamples \geq 3$,
$examples : [MAX\_MEMORY]$ all memorised examples in history sorted in descending order according to the *morph agent* criticality

1: {The 3 most critical examples are used}
2: $meanEd \leftarrow meanE_dValue(3)$
3: $meanPd \leftarrow meanP_dValue(3)$
4: $squareE_d, sumMeanE_dP_d \leftarrow 0.0$
5: $j \leftarrow 0$
6: **while** $j < 3$ **do**
7: {Compute of mean square sum}
8: $squareE_d \leftarrow squareE_d + (examples[j].valueE_d() - meanEd)^2$
9: $sumMeanE_dP_d \leftarrow sumMeanE_dP_d + (examples[j].valueE_d() - meanEd) \times (examples[j].valueP_d() - meanPd)$
10: $j \leftarrow j + 1$
11: **end while**
12: $x \leftarrow sumMeanE_dP_d / squareE_d$ {The new x coefficient}
13: $y \leftarrow meanPd - (x \times meanEd)$ {The new y coefficient}
14: $coeffX \leftarrow x \times \omega + oldCoeff \times (1 - \omega)$
15: $coeffY \leftarrow y \times \omega + oldY \times (1 - \omega)$
16: Compute the criticality of all examples in history with new coefficients
17: $sort(examples)$ {sort historic according to their criticality in descending order}

Still, with the idea to limit resources utilisation, the number of stored examples is limited to 20. This number has been experimentally chosen. These experiments are presented in chapter 5. Only the worst examples (causing the highest criticality) are kept in the history. This ensures that the *morph agent* keeps trying to improve its transformation function. The algorithm 2 details how a *morph agent* selects the examples to keep.

This behaviour enables to observe two trends: either the criticality decreases until it converges around a low value with the augmentation of observed examples, either no convergence is observed. The first trend occurs when there is a relation between both the data objective d_o and the *data agent* creator d_c. In other words, a low criticality means that the *morph agent* is able to transform the referential frame of its d_c into its d_o. For example, the *data agent* d_1 representing e_{d_4} has created a *morph agent* $ma_{4,1}$ with as d_o the *data agent* representing p_{d_1}. After several examples, the normalised criticality of $ma_{4,1}$ is 0.05 so it is able to transform the value of e_{d_4} into p_{d_1}.

However, the *morph agent* creation and behaviour have a cost on resources utilisation. Having too many *morph agents* in the module may put the LUDA module into a Non-Cooperative Situation (NCS) thus the system is no longer guaranteed to be functional. The fact that a *morph agent* is unable to transform the referential frame makes that its actions are useless for the system. Concretely, a *morph agent* that is unable to

Algorithm 2: Morph memory

Require: $nbExemple, maxExemples, newExemple, examples$: [] all memorised examples in history

{find the least critical example}
$lastExemple \leftarrow examples[nbExemple - 1]$
if $nombre < maxExemples$ **then**
  {find where to store the new example}
  $ind \leftarrow findIndMoreCrit(crit(newExample))$
  $addExample(examples, newExample, ind)$
else
  {if the new example is more critical than the least critical}
  if $crit(newExample) > crit(lastExample)$ **then**
    {remove the least critical example}
    $removeLastExample(examples)$
    {find where to store the new example}
    $ind \leftarrow findIndMoreCrit(crit(newExample))$
    $addExample(examples, newExample, ind + 1)$
  end if
end if

reduce its criticality after many examples and computation is in a NCS of uselessness, considers itself useless and to come back to a cooperative state, it destroys itself. However, to avoid inappropriate destruction because of a slow adaptation process, a *morph agent* considers itself useless only if its criticality is high and if another *morph agent* from the same *data agent* has a low criticality. This is a cooperative behaviour between *morph agents* having the same d_c. Thanks to their criticality exchange, *morph agents* can understand that their d_o's related data are not the same as their d_c. An algorithm 3 shows how a *morph agent* decides to destroy itself or not.

Algorithm 3: Morph Agent destruction

Require: $time$ {The number of adaptation process since the last time the morph agent has been in a low criticality state}
Require: MAX_TIME {The maximum of time acceptable}

if criticality > threshold **then**
  if $time > MAX\_TIME$ **then**
    ObserveOtherCriticality()
    if a neighbour has a low criticality **then**
      destroy()
    end if
  end if
end if

4.2.5 Group Agents

As different e_{d_j} represent the same d_{p_i} and can be used for the same ADP input i, a third agent type called **group agent** is introduced. It is activated at the ❹ step, after *morph agents* and *data agents* have acted. When

all *group agents* have terminated, the organisation LUDA cycle has ended. Once LUDA has ended, this one is available to give answers to the ADP.

A *group agent* is created by a *data agent* when this one is not linked to a *group agent*. A *data agent* cannot send a value (transformed or not) to the ADP by itself but it requires a *group agent*. In consequence, *data agents* want to group together and if the *group agent* includes a *data agent* whose related information is included in P_D then it can send a value to the ADP for the corresponding input i.

The first local objective of a *group agent* is to gather the maximum of *data agents* representing the same environmental information together while respecting their adequacy (or homogeneity). The adequacy of the *data agent A* toward the *data agent B* is the criticality value of the *morph agent* having as d_c the *data agent* A and as d_o the *data agent B*. If this criticality is too high (greater than a threshold), then *data agents* are not satisfied in this group and it is not stable. For example, suppose two *data agents A* and *B* having respectively the information p_{d_1} and e_{d_4} that represent the same environmental information. So, if p_{d_1} is missing (*A* is not activated), e_{d_4} could be used to replace it. It is possible if they are grouped together by the *group agent*.

The second local objective of a *group agent* is to have the most reliable value to give to the ADP when it includes at least one *data agent* related to a perceived information (P_D). This objective enables to address the noise issue by taking advantage of the redundancy of the same information.

The environment of a *group agent* is composed of all the other *group agents* in the environment, the *data agents* grouped by it, and the ADP (the only agent type to see it). It cannot see neither access *morph agents* and then rely on the value given by its *data agents* to get a value. A *group agent* can execute four types of actions: 1) **to propose one of its *data agents* to another *group agent***, 2) **to propose a merging with another *group agent***, 3) **to exclude a *data agent*** and 4) **to send a value to the ADP**. The latter is an action initiated by the ADP when it needs a value, whereas other actions are done while organising the system. All these actions are summarised in the figure 4.5.

When a *group agent* is activated it observes its internal medium to evaluate the internal organisation of its *data agents*. Each *data agent* evaluates its satisfaction by looking at its *morph agents* criticality towards the other *data agents* in the group. Then, the *group agent* aggregates those values to obtain its personal criticality, see equation 4.4. To compute the criticality, the *group agent* requires from its *data agents* the criticality of their *morph agents*. A *morph agent* gives its criticality only if its d_o is also in the group. Then, the *group agent* aggregates all the criticalities into one value. This value is normalised using the number of *data agents* in the group to make it comparable with other *group agents*. Because the first objective of a *group agent* is to gather the maximum of *data agents* representing the same environmental information, equation 4.4 is used by a *group agent* to compute its criticality.

$$crit_{ga} = \frac{\sum_{(i=0,j=1,i\neq j)}(crit(ma_{da_i,da_j}))}{nbDataAgentInGA} \tag{4.4}$$

Now that a *group agent* can evaluate the distance to its objective (through its criticality), it cooperate with other *group agents*. The first action, **proposing** a *data agent* is an easy way for the *group agent* to reduce its criticality by asking other *group agents* to incorporate non homogeneous *data agents*. To perform this action, a neighbouring *group agent* must accept this *data agent*. So, the *group agent* receiving the *data agent* proposition evaluates the impact if it accepts the proposed *data agent*. This evaluation uses the criticality given in equation 4.5. In this equation, the δ coefficient (taking its value in the range $[0.1, 2.0]$) represents the desire of the *group agent* to incorporate the maximum *data agents* possible. Lowering this parameter makes a *group agent* less likely to accept a *data agent*. So, when δ is low, the likelihood to accept a non-homogeneous *data agent* for a *group agent* is reduced. However, the time to group all homogeneous *data agents* is increased. With a high δ value a *group agent* will accept more easily new *data agents*. Once every *group agent* has computed its new criticality, the *group agent* that has initiated the proposition gives the *data agent* to the *group agent* having the best criticality with the proposed *data agent*. It has to ensure that: 1) the new criticality

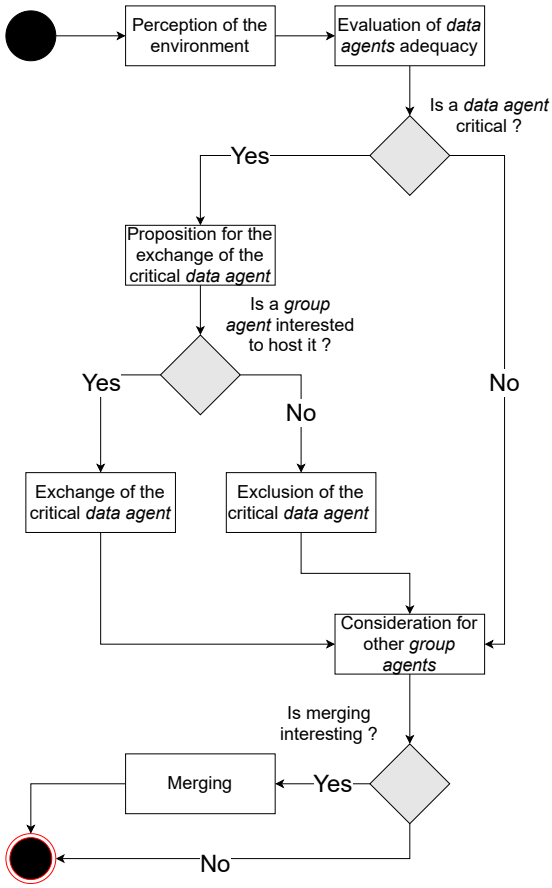


Figure 4.5: Life cycle of a *group agent*.

of the receiving one is reduced in a better way than any other or 2) the new criticality is not worse than the previous one of the receiving.

$$crit_{ga} = \frac{\sum_{(i=0,j=1,i\neq j)}(crit(ma_{da_i,da_j}) - \delta)}{nbDataAgentInGA + 1} \tag{4.5}$$

It may be possible that one *data agent* has no related *data agents*, either because these have not yet being perceived or because the ADP does not include this information in its process (i.e. gas information for an electrical car). A *group agent* must have the means to evict such a *data agent* as it is problematic. But as a lone agent, no *group agent* would accept to incorporate it. That is why a *group agent* has the possibility to expel it from the group. Doing so, the *data agent* is alone but it creates a *group agent* with only it inside. So, when a new *data agent* that represent the same environmental information is created, their two *group agents* can merge together. The figures 4.6 and 4.7 summarise the cooperation between *group agents*. In figure 4.6 the *group agent* 1 sees that the *data agent* da_2 is not homogeneous with the other *data agents* in the *group agent*. Other *data agents* in the ga_1 see their corresponding *morph agent* (the *morph agent* with da_2 as d_o) having a high criticality (crit($ma_{da_1,da_2} = 0.3$) and crit($ma_{da_3,da_2} = 0.45$)). Thus ga_1 proposes da_2 to its neighbours *group agents*. Each neighbour evaluates the impact on its criticality of incorporating da_2. The criticality of the ga_3 is the lowest and less than the threshold of 0.20 used in this example, so it incorporates da_2. The figure 4.7 represents the same initial situation but no *group agent* has a criticality less

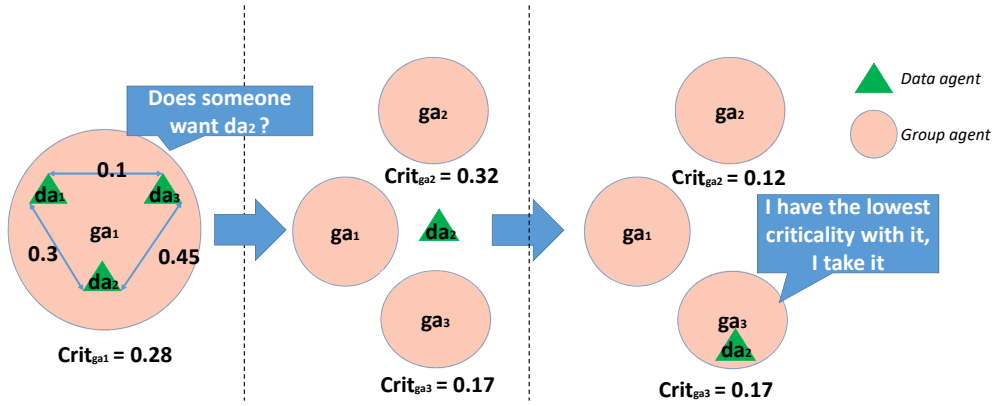


Figure 4.6: The *group agent* ga_1 has evaluated its organisation and it wants to get rid of da_2. ga_3 accepts to take it as its *data agents* criticality with da_2 is low.

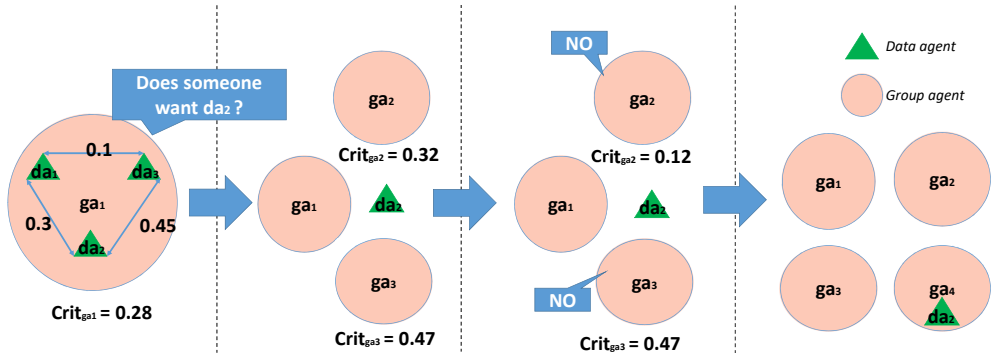


Figure 4.7: The *group agent* ga_1 has evaluated its organisation and it wants to get rid of da_2. No group agent accepts to take it due to a criticality to high. So da_2 creates a *group agent* for itself only.

than 0.20. Thus, the *data agent* da_2 is expelled and a new *group agent* is created for da_2 specifically. *Group agents* are now able to obtain good adequacy by exchanging their *data agents*. However, a situation where two different *group agents* made of 7 *data agents* representing the same information in different referential frames is likely to happen. In this case each *data agent* in each *group agent* has a criticality near 0 and the *group agent* wants to keep the *data agent*. The ideal solution is to group all these *data agents* together into a unique *group agent*, two *group agents* can merge to form only one group. The two corresponding groups initiate a negotiation to know if the merging is worthwhile. Using a similar process, they evaluate the inclusion of new *data agents* and the impact on criticality. If the resulting group is stable then the merging is made resulting in the destruction of a *group agent* in the environment.

4.2.6 Cooperative Resolution

The three types of agents have been defined, and their behaviour explained. In the AMAS approach, the overall system resolution comes from the aggregation of all simple behaviours and local interactions between agents. Table 4.1 summarises the characteristics of the LUDA module entities and environment.

The LUDA module organisation begin with the initialisation phase where only *data agents* with information from P_{D_s} evolve in the environment. Then the LUDA module receives information related to the

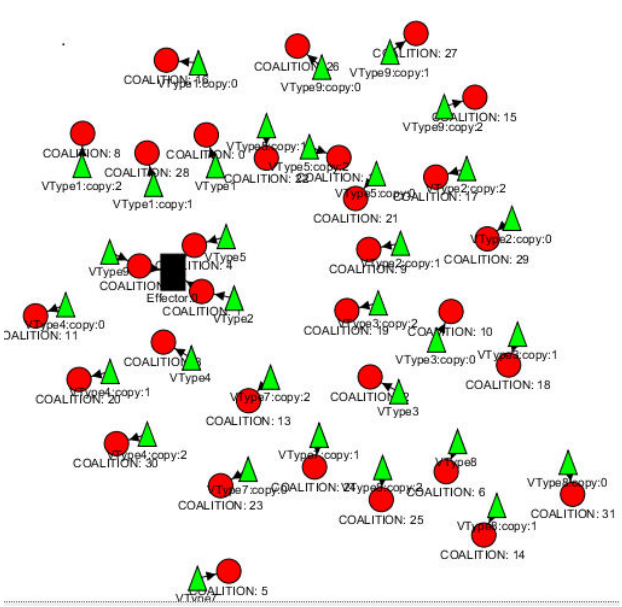
	Agent type		
	Group	Data	Morph
Local objective	To group the maximum number of data agents while keeping it homogeneous To send the most accurate value to the ADP	To be in a homogeneous group	To predicts with accuracy the value of the objective data from the value of its creator
Phase(s) of activation	Phase ❹ and outside organisation when the ADP requires it	Phases ❶ and ❸	Phase ❷
Entities perceived from the environment	ADP Group agents Data agents	Data agents Own morph agents Own group	Morph agents with the same creator The data agent creator The data agent objective
LUDA objective affected by it	Prediction of a value Increasing accuracy	Replacement of missing information	Prediction of a value
Skills	Proposition of a data agent Expulsion of a data agent Merging with a group agent Sending a value to the ADP	Creation of morph agents Observation of the environmental information	Production and adaptation of a transformation function
Cooperative behaviour	Incorporation of a data agent Self-destruction for merging	Leaving a group agent Joining a group agent	Self-destruction

Table 4.1: Summary of the characteristics, behaviours and objectives of the three types of agents.

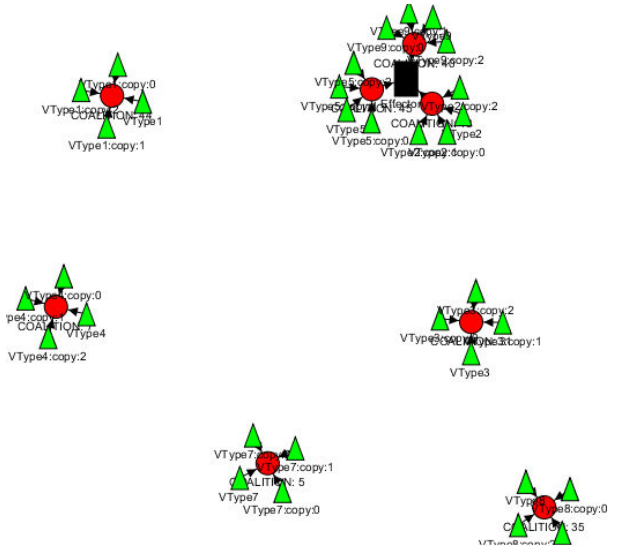
communication of other CAVs. For each one of these communications, a *data agent* is created. Then *data agents* create a *morph agent* for each other *data agents* in the environment. Once all agents are created they follow their local behaviour. *Morph agents* improve their transformation function or they destroy themselves. *Data agents* group in *group agents* if they agreed. The successive merging of *group agents* reduce their number. The system organisation convergence is reached when the number of groups is equal to the number of different environmental information in the system. Moreover, *data agents* must only be in a group where they are all related together.

Figure 4.8 shows an example of organisation with 8 environmental information and each exists in 4 different referential frames ($|P_D| = 8$ and $|E_D| = 24$). In this figure, the ADP (represented by a black rectangle) takes 3 P_{D_s} (only 3 to facilitate the visualisation). An input is represented by an arrow between a group agent (red circle) and the ADP. For visualisation purpose, *morph agents* are not represented. The left side figure 4.8a represents the initial state where *data agents* have no knowledge about each others. *Data agents* are alone in their personal group. With a continuous adaptation, *morph agents* improve their function and the organisation is modified until the system convergence (right side figure 4.8b). The find number of *group agents* is eight, each with 4 *data agents*.

It illustrates the self-organisation mechanism providing an accurate structure for the module. This structure enables the use of information in different referential frames by the ADP after their values have been transformed.



(a) Groups Distribution at the first organisation cycle. Data agents are represented with a green triangle and they are alone in their group (represented by a red circle).



(b) Groups distribution after several organisation cycles. After several adaptations realised by data agents and their morph agents, the number of groups is reduced to the number of different environmental information (8 in this example).

Figure 4.8: Example of the self-organisation of the LUDA module. In this example, three P_{D_s} are used by the ADP to compute the result.

4.2.7 Sending Inputs Value to the ADP

This section presents the last action a *group agent* can do. This action is not part of the organisation process of agents but is made available thanks to it. With the use of the three actions (proposing a *data agent*, proposing a merging, and excluding a *data agent*) and *data agents* as well as *morph agents* behaviours, a *group agent* has access to several instances of the same environmental information. Information redundancy is often to be avoided, because it produces no new knowledge. But here, as the environment is considered noisy and the **real** value is altered by sensors quality and errors, fusing multiple sources of the same information into one has several benefits:

- When an information is missing, it can be replaced with another one (or several).
- A sensor has a margin of error that depends on its quality and the data collected. This error may vary but with several instances of the same data, a more accurate value can be produced.
- In a critical system, the use of redundancy allows protecting the system (here sensors) from failure (often a 3 redundancy is used). However, it has a cost on resources.
- Detecting malevolent or bad entities using coalitions. Similar to the anomalies detection, combining several sources gives a way to exclude problematic entities from the cooperation process. For example, three CAVs exchange the same environmental information about a road that is empty of vehicles. Then a fourth CAV exchanges that the road is not empty because doing so, it gains the priority. The first three CAVs have learnt that the fourth one cannot be trusted so they ignore its communication.

The fourth action a *group agent* can do is **proposing a value** for an input i of the ADP. When every input has a value, the ADP can compute a result that corresponds to the system global objective. This action is only available for *group agents* having a *data agent* representing an information from P_D. So, a *group agent* has another objective: to minimise the difference between the proposed value, and the real value perceived later by the CAV agent. With access to its grouped *data agents*' value a *group agent* fuses them into a more accurate one. Excepted the *data agent* related to information from P_D, each other gives its prediction of the latter's value to the *group agent*. For example, two *data agents* representing respectively p_{d_1} and e_{d_4} are grouped by a *group agent* ga_1. It has to send a value to the ADP, it takes value from the two *data agents*. For e_{d_4} the value taken by the *group agent* is the value transformed by the *morph agent* $ma_{4,1}$ (with e_{d_4} as d_c and p_{d_1} as d_o). From these two values, ga_1 computes a new value to send to the input i_1.

A *group agent* associates a weight w_j with each active *data agent* in its group. The weight of a *data agent* represents the impact of the *data agent*'s prediction on the *group agent* proposed value. The proposed value is the result of a weighted sum divided by the number of active *data agents* (see equation 4.6). Using this equation means that even with missing information, the *group agent* is able to provide a value thanks to the redundancy.

$$value_{ga} = \frac{\sum_{i=0} w_i * da_i}{\sum_{i=0} w_i} \tag{4.6}$$

Once all the inputs of ADP have a value, the ADP computes a decision, and then the *group agent* receives a feedback from the ADP. This represents the capacity of an ADP to have access to the **true value** after deciding an action and it gives the information back to LUDA. The feedback is obtained because the ADP can understand the values it should have had after doing the action. The feedback is considered as the ideal value that should have been given for the input. For example, for the input i_1, representing the relative position of a hidden pedestrian, the value 12 has been given by a *group agent* ga_1. After deciding a behaviour, it perceives the true position which was 11,6. This value is given to ga_1.

From this feedback, a *group agent* gets the information about the ideal value and adapts the weight according to it. More accurate *data agents* are rewarded with a weight increase while the *data agents* less accurate see their weight decreases. In addition, a *group agent* can use some trends in the data to compensate for errors. For example, a sensor has the trend to always overestimate a value while two others underestimate it. By increasing the weight of the first instead of the two others, the underestimation is compensated. So, a *group agent* increases the weight of *data agents* whose values would impact positively the *group agent* value if its weight was higher. Equation 4.7 is used by a *group agent* to modify the weight of its *data agents*. The weight cannot be reduced to 0 because if the corresponding *data agent* is the only one available, it is still used to replace missing information. The fields use in the equation 4.7 are:

- V_{ga} the *group agent* value.
- V_{ADP} the ADP true value for the input.
- α, the increase when the value of the *data agent* of index i is better than the one computed by the *group agent*.
- β the decrease when the value of the *data agent* of index i is worse than the one computed by the *group agent*.
- γ the increase value when the ADP value is between the V_{ga} and V_{ADP}.

After many exploration experiments, the values $\alpha = 0.1$, $\beta = 0.05$ and $\gamma = 1/30$ have been chosen for the experiment part.

$$w_i = \begin{cases} w_i + \alpha \text{ if } |V_i - V_{ADP}| < V_{ga} - V_{ADP}| \\ w_i - \gamma \text{ if } |V_i - V_{ADP}| > |V_{ga} - V_{ADP}| \\ w_i + \beta \text{ if } V_{ADP} \in]V_{ga}, V_i[\text{ or } V_{ADP} \in]V_i, V_{ga}[\end{cases} \tag{4.7}$$

Noise is taken into account by *group agents* and not by *morph agents*. Indeed, *group agents* use a reward strategy that increases the impact of *reliable data agents* and decreases the less reliable ones. Highly noised information has a small impact but it is still used when there is no other information available for an input.

4.2.8 Conclusion on the LUDA module

The LUDA module has been designed to enable an agent to understand the information in another referential frame communicated by other agents, thus allowing cooperative behaviours. Evolving in a dynamic, large, and noisy environment the module has the challenge of transforming information by estimating the value from others while being light computational.

The LUDA module is an AMAS within an agent. This approach has been chosen for its quality in openness and scalability. This is motivated by the CAV agent environment that is non-finite and continuously evolving. The module is composed of three types of agents: *data agent*, *morph agent*, and *group agent*, each with their local objective and local view. After several steps, the module can provide an accurate estimation of missing information. This is achieved thanks to the self-organisation of all agents together; an emergent solution (a value for an input) is collectively found. Moreover, it can provide an information with increased accuracy thanks to the redundancy of the same information.

The validation of the solution is presented in chapter 5. The LUDA module is compared to existing data estimation methods by evaluating the accuracy of the prediction. Then each, characteristic is explored to validate the **scalability** and **openness** of the solution. Now that the CAV agent can understand other agents' communication, the second contribution of this thesis concerning an optimisation module is presented.

4.3 Distributed Communication Optimisation in a Fleet of CAVs

The problem stated here addresses the communication optimisation for the cooperation of a fleet of CAVs. With the development of smart cities and ITS the number of devices has significantly increased. Numerous activities from collecting data to city management are foreseen and require dedicated connected devices. Those physical devices have common characteristics: 1) their perception range is limited, 2) their communication range is limited and 3) they are dynamic either by their mobility (i.e. CAV) or by their activity (i.e. sensors).

The fleet of CAVs context considers CAVs as social entities exchanging information. By doing so every individual in the fleet has an increased comprehension of the world it evolves in. Many state-of-the-art works use V2V communication to exchange information to improve CAVs' behaviour. Instead of using only their perception, CAVs can rely on the perception of other CAVs. A fleet can share enough information about the environment to create a local shared visualisation of their environment [5]. As CAVs are becoming more and more complex, using an increasing number of sensors, the amount and size of information are growing. However, every perceived information is not useful to communicate, a selection must be done. The same object, for example, a pedestrian, can simultaneously be perceived and communicated by several CAVs. This perception quality depends on many factors either from the one perceiving (sensors, position, light,...) which impact the reliability or from the one receiving (referential frame, goal,...) which reduces the overall usefulness of communication. This information is useful for other CAVs who cannot perceive it (i.e. a truck blocking the camera vision). Knowing this, CAVs share the information with other nearby CAVs.

From the LUDA module description in the previous section, CAVs can exchange information that may be used by each other. They also have the possibility of such a locally shared visualisation. However, this shared visualisation comes with its share of issues:

- **The amount of information possible to share is huge and constantly increasing**. The development of the 5G connection enables the exchange of a large amount of data within a short time. In addition, by the time fleets of CAVs exist, the 6G is likely to happen which is foreseen to enable even more data transfer. However, on the CAV's side, some issues can be mentioned. A CAV is a critical system evolving in real-time and responsible for roads user security. With an increasing amount of communications, a CAV may be limited in terms of information processing.

- **Every information is not useful to share**. For example, sharing the position of a crossing pedestrian is interesting but only if the crossed road is on the receiving CAV's trajectory. If the CAV is not on the crossroad, this information is useless to exchange.

- **The shared information must be as light as possible**. An image of quality 4k is 9 times bigger than an image in 720p and costs all the more in terms of network traffic and processing at the reception.

- **Learning another entity and its goals is not efficient**. Learning preferences of a particular entity requires time and multiple exchanges. ITS context is highly dynamic due to the mobility of the entities composing it. At the time the preferences are learned, the entity has already left the perception range.

- **A redundant information should be shared only if it brings an improvement**. Many CAVs are likely to perceive the same information about the environment. They have to only share it if it brings new knowledge for a CAV in the fleet.

These issues clearly show a need for **optimising which information has to be exchanged** within a fleet of CAVs. The data processing time needed for a CAV may be limited and the exchange of the totality of information is not possible. Indeed, suppose n CAVs each perceiving m information, and v_m the volume

of an information. If the totality of information is exchanged, the total volume vt of shared information is $vt = n \times v_m \times m$. This limits the scalability of the solution.

The problem addressed here is how agents decide whether or not to communicate any information. Associating an information with an agent can be modelled as an allocation problem to maximise the overall information usefulness received by each agent. This allocation problem is framed by processing constraints that limit the number of information that can be received. Given agents' dynamicity, a dynamic solution is needed and centralised optimisation methods are not suitable in this context to avoid the loss of privacy.

Because of the naturally **distributed** control of each CAV, constraints framing the problem and the objective of maximising a result, the problem formalisation uses the well-known DCOP framework. Each agent has its constraint related to its computation capability (the number of information it can process). An agent perceives information from its environment with a quality; the objective is to maximise the global quality (i.e. the sum of quality) of information exchanged by CAVs.

This section presents the second AMAS-based module called CODCOP that enables a CAV to optimise which information to send to other CAVs in the fleet. To cope with the challenges of **dynamic**, **openness**, **scalability** and **privacy**, the CODCOP module design is based on the AMAS approach. First, the DCOP formalisation of the problem is presented, followed by the challenges description. Then, the agents design is described, followed by the details of the cooperative approach used by agents to address the problem. After a description of the emergent resolution is given before concluding.

4.3.1 CAV Problem Description

Let's suppose road traffic in an urban area. Some vehicles of the fleet are CAVs equipped with devices enabling V2V communications. Through the VANET protocol, a CAV can reliably communicate up to 200 meters. Sharing information about their common environment is a way for the CAVs to improve their understanding of the environment and thus their actions choice. However, road traffic is a complex system, involving many entities with a large amount of information (position, speed, vision, traffic-related information,...), some of them being the same but differently perceived. For example, the speed of an object perceived by a CAV agent is 50 km/h. This CAV agent also received by another agent a speed equal to 50,65 km/h. The second information is more accurate, thus inducing a better decision for the Autonomous Driving Process (ADP). But every information cannot be shared, suppose that each CAV communicates all the information it knows, the number of communication to be processed will likely exceed the capacity for a CAV to process them. To enable information sharing, CAVs need to select the useful information to be sent without overloading another CAV. Moreover, road entities are mobile by nature and the overall system is highly subject to dynamic changes.

To resume, the problem addressed here is the following: several CAVs, spatially positioned, have information that may be useful to other CAVs situated within their range of communication. A CAV does not know among all its information, the subset that is useful to share; it has to decide without this knowledge. This is enforced by the dynamicity of the stated problem which is a major obstacle to solving the problem by preventing a CAV to learn each other CAV internal information. CAVs are constrained by their processing capabilities and their personal goals that have to be kept private.

4.3.2 Distributed Constraint Optimisation Problems Definitions

The DCOP is a powerful framework to address optimisation problems with constraints. The resolution is done by distributing the overall problem among several agents. DCOP has been widely studied and used in many application fields [39]. The motivation for using this framework lies in its strong properties. First, agents use local messages exchange to find an acceptable solution satisfying every agent. It respects the distribution

control of CAVs and local goal and preferences. Second, the domain structure is encoded within constraints, thus allowing to address hard computational problems as seen in chapter 3. This is well-adapted to the stated problem as each CAV possesses a part of the problem and its actions are the motor for the overall resolution. A DCOP is usually represented by the 5-tuple $< A, \chi, D, F, \mu >$:

- A = $\{a_1, ..., a_{|A|}\}$ is the set of agents; in the context of fleet of CAVs, an agent is a CAV.
- $\chi = \{x_1, ..., x_n\}$ is the set of all variables owned by agents; in the addressed problem, these variables are limited to the information **perceived** by CAVs agents.
- D = $\{D_{x_1}, ..., D_{x_n}\}$ is the set of finite domains such that variable x_i takes values in D_{xi} = $\{v_1^i, ..., v_k^i\}$. In this problem, all variables take their value in the finite domain $D = \{0, 1\}$.
- C = $\{c_1, ..., c_m\}$ is a set of **soft constraints**, where each c_i defines a cost $\in \mathbb{R}$ for each combination of values to the variables in its scope (a constraint is only known by agents involved at the initialisation).
- $\mu : \chi \longrightarrow A$ is a function that assigns the control of each variable to an agent.

A DCOP solution is a complete assignment σ that minimises a global objective function $F(\sigma)$. $F(\sigma)$ aggregates the individual cost functions such as $F(\sigma) = \sum_i c_i$. A solution is optimal when it minimises (or maximises depending on the objective) F.

4.3.3 Problem Characteristics

The DCOP associated with the problem of optimising information between mobile and intermittent devices has the following characteristics: an agent has its perceptions of the environmental information. If we consider two agents, as their sensors are different (type, position, etc.) these agents perceive the same object but with different values.

- We then note $x_{i,j}$ the variable x_i perceived by the agent j. Thus $x_{i,0}$ and $x_{i,1}$ represent the same information in the environment from different points of view. Each variable has its values in the domain D = $\{0, 1\}$ representing the fact that the information is communicated (1) or not (0).
- Each $x_{i,j}$ has an **utility** function that represents the utility of this variable for the agent (for example $utility(x_{0,0} = 7.5)$. Thus the cost of a variable assignment depends on other variables related to the same information.
- The cost used for the objective function (noted $cx_{i,j}$) is the highest among all variables representing the same information received and/or perceived by the agent and follows equation 4.8. By default, all variables have 0 as value, the information is not communicated. When a CAV decides to share an information, the variable's value is 1. So the utility of this variable is taken into account in the constraint. Note that at the initialisation, a constraint $cx_{0,1}$ has only the agent's variable ($cx_{0,1} = max([x_{0,1} \times utility(x_{0,1}))$. When receiving communication, the matching variables are added to the soft constraint. For example, the agent ag_1 has the soft constraint $cx_{0,1}$ for the variable it perceives $x_{0,1}$. A neighbour agent, ag_2, gives the value 1 to the variable $x_{0,2}$ (so the CAV ag_2 sends the information represented by the variables $x_{0,2}$). In consequence, the soft constraint $cx_{0,1}$ of ag_1 is computed with $cx_{0,1} = max([x_{0,1} \times utility(x_{0,1}), x_{0,2} \times utility(x_{0,2})\})$
- The global objective function of the fleet of CAVs is to maximise the utility of transiting information. The function to be maximised is the sum of costs for all constraints for all agents (eq.4.9).

$$cx_{i,j} = max([x_{i,0} \times utility(x_{i,0}), ..., x_{i,|A|} \times utility(x_{i,|A|})] \tag{4.8}$$

$$objective = \sum_{j}^{|A|} \sum_{i}^{n} cx_{i,j} \tag{4.9}$$

With each variable $x_{i,j}$ is associated a weight $w_{i,j}$ representing the size of the information (i the environmental information; and j the perceiving agent). Let be $max\_process_j$: the maximum processing capacity of the agent ag_j can process. Thus the sum of communications of all its neighbours must not exceed its capability. This is represented by the hard constraint (equation 4.10), adding $-\infty$ to the global objective if the constraint is not respected.

$$ccom_j = \begin{cases} 0 \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l} \times w_{i,l} \leq max\_process_j \\ -\infty \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l} \times w_{i,l} > max\_process_j \end{cases} \tag{4.10}$$

The variables cost is the sum of the utility for all agents receiver of this variable. It is important to note that in this problem, the utility of an information is different according to the receiving agent. For example, utility($x_{0,0}$) = 7.5 for ag_1 and utility($x_{0,0}$) = 5 for ag_2).

In a classical DCOP, constraints are symmetric, signifying that the cost induced by an assignment is the same for all agents concerned by it ($cx_{i,1} = cx_{i,2} = cx_{i,3}$, etc.). In this problem the utility of a variable can differ between agents, agents can have a different cost for the same assignment ($cx_{i,1} \neq cx_{i,2}$). This problem is then considered as an Asymmetric DCOP (called ADCOP). In addition, because of the agent's nature (mobile and/or intermittent), the problem changes over time as variables' domains and constraints' cost functions may change, this problem is also a Dynamic DCOP (called Dyn-DCOP). Solving a Dyn-DCOP means finding a solution for **each** DCOP in a sequence of several successive different DCOPs. To conclude, this problem is an A-Dyn-DCOP one. To follow the formalisation, a tuple is added to the 5-tuple called T including the set of modifications of the problem. Formally T is expressed by: $T = \{t_0, t_1, ..., t_{final}\}$ with t_l a set of modifications when the time l is reached. There are three types of modifications that cause the problem to be changed:

- The arrival of an agent in the system. An agent comes with its part of the problem (variables, constraints, and range). Moreover, the arrival of an agent modifies its neighbours' constraints as it can share new variables. For example, a CAV enters the fleet.
- A constraint modification. Local sub-objectives of an agent can evolve, so local preferences are modified as well as its constraints structure. For example, after letting a pedestrian pass, a CAV wants to cross the intersection, it is more interested in information about it than when it was stopped.
- Variable appearance or disappearance. For several reasons, an agent can be unable to perceive an information it previously could. It is concretely modelled as the variable has its value put at 0 without the possibility to change the value until the variable is perceived once again. For example, a pedestrian is hidden by a truck from a CAV point of view, the information has disappeared.

Usually, in DCOP, an agent is responsible for one variable. Here this assumption is not met as an agent possesses as many variables as it can perceive. This is possible by the "simplicity" of the variables domain, which is binary (0 or 1). The second reason is to be close to the application domain (one CAV = one agent). Following the common DCOP assumption, an agent CAV would include an agent for each perception. The benefit of distributing in one agent is small as the variables domain is low. It is assumed that communication

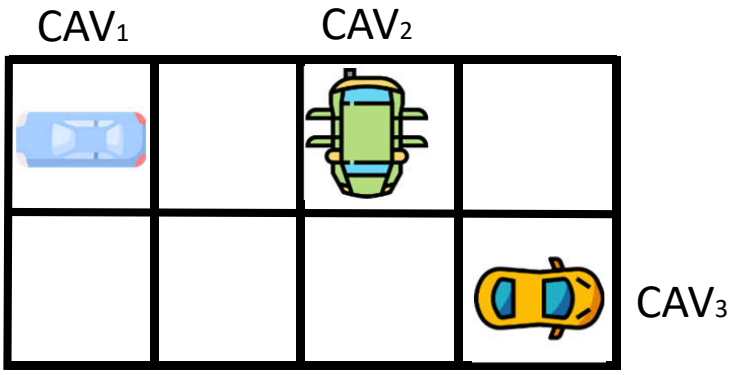


Figure 4.9: Example of a situation with 3 CAVs. Each one sees a local part of its neighbourhood and communicates within a range of two cells.

is reliable, understood and non-malevolent because not in this AMAS scope. The reliability assumption one is not in the thesis scope and is from another domain. The understanding assumption can be made thanks to the LUDA module previously described. And the last is a part of this thesis' perspectives which are detailed later.

4.3.4 Graphical Representations

DCOPs are often represented using graphical models, called constraints graphs. This helps to understand the problem's characteristics and the relations between agents. Figure 4.9 is an example illustrating the addressed problem. It considers a fleet composed of three CAVs (CAV_0, CAV_1 and CAV_2), each with a range of communication such that CAV_0 and CAV_1 are connected, as well as CAV_1 and CAV_2, but not CAV_0 and CAV_2. Each CAV perceives a subset of environmental information:

- $P_0 = \{x_0, x_1, x_2\}$.
- $P_1 = \{x_0, x_1, x_3, x_4\}$.
- $P_2 = \{x_0, x_1, x_2, x_3\}$.

In addition, each ADP of the three CAVs requires a subset of environmental information to make a decision (noted ADP_i for the ADP of CAV_i):

- $ADP_0 = \{x_0, x_1, x_2\}$.
- $ADP_1 = \{x_0, x_1, x_2, x_3\}$.
- $ADP_2 = \{x_1, x_2, x_4\}$.

This example illustrates that an ADP may require an environmental information that is not perceived by the CAV (the ADP_2 requires x_4 but it does not perceive it. From this example, a constraint graph is created for graphical representation.

Figure 4.10 is the graphical representation of the figure 4.9 with 3 agents (a_0, a_1, a_2) representing respectively CAV_0, CAV_1 and CAV_2 (computational constraints are not represented for visualisation purposes). Each agent is represented by a coloured area. Within each area, orange circles are agent's variables and orange

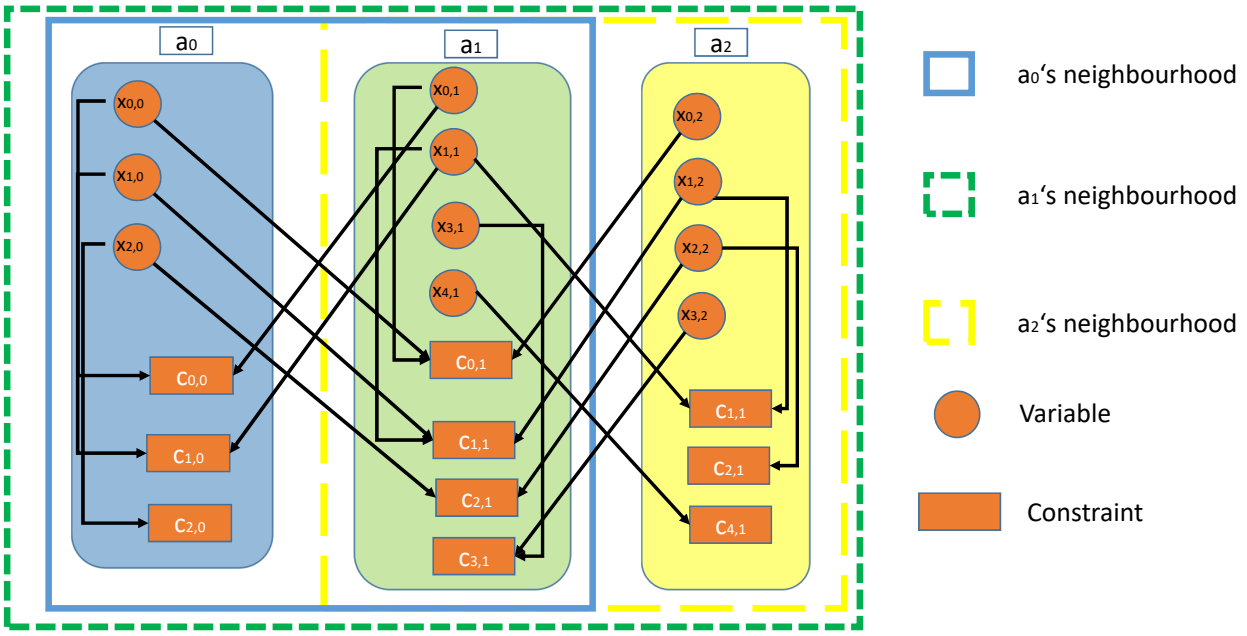


Figure 4.10: Constraint graph example of the communication optimisation problem.

rectangles are the soft constraints previously described. Black arrows represent the link between a constraint and the concerned variable in all the neighbour agents. For example, constraint $c_{0,0}$, is related to variables $x_{0,0}$ (the variable representing the environmental information x_0 perceived by ag_0) and $x_{0,1}$ (the variable representing the environmental information x_0 perceived by ag_1). For visualisation purposes, $ccom$ is not represented as it is linked to every other variable from an agent's neighbour.

Large empty rectangles represent an agent's neighbourhood. In this figure, the blue line represents a_0 neighbourhood, while the yellow one represents the a_2 neighbourhood and the green dotted one for a_1. a_0 neighbourhood includes a_1 but not a_2. Consequently, variables costs of a_0 are only linked to variables of a_1. a_2 does not have a_0 in its neighbourhood, but only a_1. Thus the cost of a_0's variables only takes into account those from a_1. In this example, a_1 knows both a_0 and a_2, thus being impacted by both but a_0 and a_2 cannot communicate together.

From this representation, several observations can be made. First, an agent can have a soft constraint linked to a variable that it does not perceive, as it can be seen with $c_{4,1}$ for a_2 only linked to the variable $x_{4,1}$ owned by a_1. Secondly, it is possible that an agent does not use one of its variables but it may be useful to another like $x_{3,2}$ and $c_{3,1}$. Finally, the neighbourhood has a strong impact as a larger neighbourhood means that constraints became bigger and/or numerous.

4.3.5 Optimisation under Communication Constraint Challenges

To obtain an organisation that maximises the global objective function, some criteria are to be taken into account. The first criteria is the exchange of private information. By distributing the problem over several agents, none of them has complete knowledge about the problem. One drawback of distributing the problem among agents is that agents have a limited view and a partial knowledge of the problem's constraints. The value choice of a variable is made without knowing the variable effect on the neighbour agents constraints. This can be problematic to find an accurate assignment in the context of asymmetric problems. Indeed, when an agent decides a value for a variable without having access to every constraint impacted by it, it may worsen

the previous solution instead of improving it. This can easily be seen with the MGM algorithm [61] which in a normal context ensures a monotonous improvement of the solution but no longer true when confronted with asymmetric problems. This is why a proposed algorithm extension called MCS-MGM [51] follows a strategy of sharing information about the constraint but doing so leak private information. This is problematic for three reasons:

- Such information might be not easily accessible but only an answer from another system. Relying on the constraint exchange may not be possible in a specific context. For example, the internal structure of the CAV's ADP is hidden so the module cannot access the *soft constraint* structure, thus it cannot share it.
- Sharing a constraint has a cost on computation and communication.
- Data and lately meta-data gives more and more information about the characteristics of an entity. Tools and techniques are improved, and from a simple information, many accurate assumptions are extracted. An entity may not want such information being leaked. For example, sharing its destination enables the tracking of the person in real-time.

For these three reasons, **privacy** is a characteristic that must be respected at the maximum. It is the first criteria in addition to the maximisation of the global objective function.

The second criteria is **time-related** due to the dynamic nature of the addressed problem. The system must be able to provide an accurate assignment soon after modifications are encountered. Because it is not possible to anticipate when a modification will be encountered, an emphasis on the time to assign a value to variables is made. The second criterion is to minimise the computational time between two assignments.

To summarise, the main goal is to provide an assignment that maximises the overall usefulness of communication in the system. To evaluate the solution, two metrics are taken into account: the time that needs to be minimised and the privacy loss that also needs to be minimised. With these characteristics in mind, the proposed solution is described in the following section.

4.3.6 Cooperative DCOP (CODCOP)

The section presents the CODCOP module based on the AMAS approach for solving the A-Dyn-DCOP under communication constraints presented in section 4.3.1. The variables domain includes two values: 0 and 1. It is important to note that sending a message is the resulting action of an agent deciding to give the value 1 for one of its variables (according to the DCOP framework). Conversely, stopping sharing a message is the resulting action of an agent giving the value 0 to the corresponding variable. The proposed solution has been designed according to the privacy, openness, scalability and dynamic characteristics previously described.

Environment

The CODCOP module environment is the social environment of the CAV agent. It includes all CAVs equipped with a CODCOP module within range of communication. Formalised with the DCOP framework, the environment is a representation of the overall problem to solve. According to Russell and Norvig [128], the environment possesses the following characteristics:

- **Inaccessible:** agents do not have (and should never have) access to other agents' internal structures. They do not perceive the problem globally.

- **Indeterministic:** agents have access to their constraints which represent the result of the agent CAV's action thanks to communication. However, they do not have access to the result of the chosen communication.

- **Dynamic:** problem modifications happen without the agents' action.

- **Continuous:** an agent has its own set of variables and the number of agents is constantly evolving. Consequently, it is not possible to consider it as a finite environment.

Agent characteristics

Characteristics of a CAV agent are those presented in section 4.3.1 and reminded hereafter. An agent can perceive from its environment a certain amount of information called variables. A variable has a label built with the information name and the agent id (i.e. $x_{0,5}$ is the information 0 perceived by the agent 5). In addition to perceived variables, an agent has access to variables communicated by agents of its neighbourhood. The neighbourhood agents regroup all other agents within their communication range defined at the agent creation but can evolve through time. In addition, the communication range is limited with a highly open world. Two objectives are taken into account: to only share useful information that produces new knowledge about the situation for a receiving agent; to not overload another CAV with too much information. If an agent is unable to process everything it receives, then the agent sharing an information not processed has seen its action being **useless**.

For every perceived variable, an agent possesses an associated constraint following equation 4.8 from section 4.3.3. This constraint is not entirely available for the agent as it does not know the true impact of neighbours' variables. From the variables available to the agent, it computes a cost for all its constraints which is added to the global objective. Each agent is responsible for a computation constraint, as stated by equation 4.11, where $max\_process$ is the maximum processing capacity. This value is given at the agent creation. This computation constraint is the main restriction in the system. Formally, it is considered as a *hard constraint* which is concretely represented with an ∞ loss.

$$ccom_j = \begin{cases} 0 \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l} \times w_{i,l} \leq max\_process_j \\ -\infty \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l} \times w_{i,l} > max\_process_j \end{cases} \tag{4.11}$$

Agent Initial Behaviour

An agent entering the system possesses the previously described characteristics including perceiving variables and constraints. The agent's goal consists in maximising all its constraints cost. However, when entering the system, an agent does not have any knowledge about non-perceived variables (for example, ag_1 does not know $x_{0,3}$ exists). Using the DCOP framework, the knowledge about neighbours variables is included in the constraint definition. However, in this problem, agents do not have access to this knowledge. An agent only knows that its neighbours may have an useful information to share. An agent's initial behaviour consists in sending the most useful information without exceeding the maximum processing capacity of its neighbours. The maximum processing capacity is not shared with neighbours with regards to **privacy** and **dynamicity**. Also, the neighbourhood is not shared due to the dynamic nature of the problem. The exchange of this information is not suitable (the neighbourhood should be exchanged each time an agent arrives or leaves). Agents need another way to know how to be cooperative.

Criticality

As stated in chapter 3.5, the problem resolution comes from the agents' cooperation by lowering the criticality. The criticality allows to evaluate the distance between an agent and its goal. Here, an agent's criticality is the sum of 3 sub-criticalities, each one for a sub-goal:

- 1) the desire to obtain more useful information (neighbours' variables increasing the cost).
- 2) the desire to reduce the number of received messages (number of available neighbours' variables).
- 3) the desire to reduce the number of useless received messages. A received message is considered useless when receiving it does not increase the total cost for the agent.

Those three sub-criticalities follow a linear function going from 0 (minimum criticality) to 100 (maximum criticality). To compute the sub-criticality 1, an agent uses the number of useless messages received compared to a threshold. While the number of useless messages is low, the agent considers that all useful messages are not yet sent. Equation 4.12 is used by an agent j to compute the first sub-criticality (called crit1 in equation).

$$crit1_j = \begin{cases} 0 & \text{If } u\_v >= min\_threshold_1 \\ 100 & \text{If } u\_v <= max\_threshold_1 \\ a_1 * \frac{u\_v - max\_p_j \times min\_threshold_1}{max\_process_j} \times 100 + b_1 & Otherwise \end{cases} \tag{4.12}$$

With u_v the volume of useless messages received and $max\_process_j$ the maximum processing capability (defined in in section 4.3.3). This first sub-criticality is related to the utility constraints of the agent. $min\_threshold_1$ (respectively $max\_threshold_1$) is a threshold of the number of useless messages for which the criticality is minimum (respectively maximum). Those thresholds are defined at the initialisation of the system and can be subject to calibration according to the problem characteristics.

Coefficients a and b are the linear coefficients, computed according to the minimum and maximum thresholds. These coefficients are computed for each sub-criticalities according to the threshold values. The second sub-criticality, called $crit2_j$ (equation 4.13) is computed according to the number of received messages compared to max_process. This criticality allows to ensure that the related $ccom$ is and will not be violated.

$$crit2_j = \begin{cases} 0 & \text{If } message\_volume < min\_threshold_2 \\ 100 & \text{If } message\_volume > max\_threshold_2 \\ a_2 * \frac{message\_volume - max\_process_j}{max\_process_j} \times 100 + b_2 & Otherwise \end{cases} \tag{4.13}$$

The third sub-criticality, called $crit3_j$ (eq.4.14) is computed according to the number of useless messages received compared to the $max\_process$. This criticality allows removing useless messages.

$$crit3_j = \begin{cases} 0 & \text{If } u\_v <= min\_threshold_3 \\ 120 & \text{If } u\_v => max\_threshold_3 \\ a_3 * \frac{message\_volume - max\_process_j}{max\_process_j} \times 100 + b_3 & Otherwise \end{cases} \tag{4.14}$$

As a consequence, the three sub-criticalities represent opposed sub-goals (asking for more or fewer variables from neighbours). Those three criticalities are then aggregated to be the agent's criticality (equation 4.15) representing the state of the agent which is shared with its neighbours.

$$criticality_j = crit1_j - crit2_j - crit3_j \tag{4.15}$$

In this way, the required action of an agent to reduce the criticality is correlated to the criticality sign. If the criticality is positive the agent means that it wants to receive more messages, if the criticality is negative, it wants to receive less messages. Thus, only one criticality value is exchanged instead of 3. The thresholds values used in this work are the following:

- $min\_threshold_1 = 15\%$ of useless messages received
- $max\_threshold_1 = 0\%$ of useless messages received
- $min\_threshold_2 = 90\%$ of maximum processing capacity
- $max\_threshold_2 = 100\%$ of maximum processing capacity
- $min\_threshold_3 = 10\%$ of maximum processing capacity
- $max\_threshold_3 = 30\%$ of maximum processing capacity

These values have been chosen for four motivations.

- 1) $crit2$ must increase faster than $crit1$ to avoid the violation of the hard constraint.
- 2) $ccom$ must never be violated for long. With the exception of dynamic changes that may have as consequence to violate the hard constraint (for example, 10 new agents arrive in the neighbourhood, increasing the amount of communication).
- 3) When maximum processing capacity is low, $crit2$ does not impact the agent criticality.
- 4) As agents try to send only useful messages, the only moment where crit3 $>$ crit1 is when there is a dynamic change. Choosing the 120 value in equation 4.14 allows to fasten the adaptation to changes. Indeed, as long as $crit3$ sees its threshold exceeded then $crit1 < crit3$, thus allowing to remove useless messages.

Neighbourhood

Real devices can only communicate within a range related to their equipment. In this work, the agents' neighbourhood gathers all agents within their communication range. We assume that every agent possesses the same communication range. Thus an agent does not take into account a message from one that is not in its neighbourhood. All communication is done in broadcast in the agent's neighbourhood. The environment is represented with a grid and each cell can be associated with only one agent (fig.4.11). An agent entering the environment has at least one associated neighbour to ensure connectivity with other agents.

Cooperation

In CODCOP, agents have no other solution than cooperating. The cooperation is based on the local representation of the utility of exchanging an information. As a private information, the utility must not be shared to comply with the **privacy** challenge. In CODCOP, an agents only share its criticality, to inform its neighbours about its state. This is the only shared information about its state. In CODCOP an agent has to help the most critical agent in its neighbourhood. Thus, it compares the absolute value of all its neighbours' criticalities and then acts according to the worst one. In this problem, if the highest criticality is negative, it reduces its communication, whereas, if positive, it increases it. Moreover the agent must have the capacity to share a useful message to take its decision (i.e. an agent with no more messages to send will not increase the number

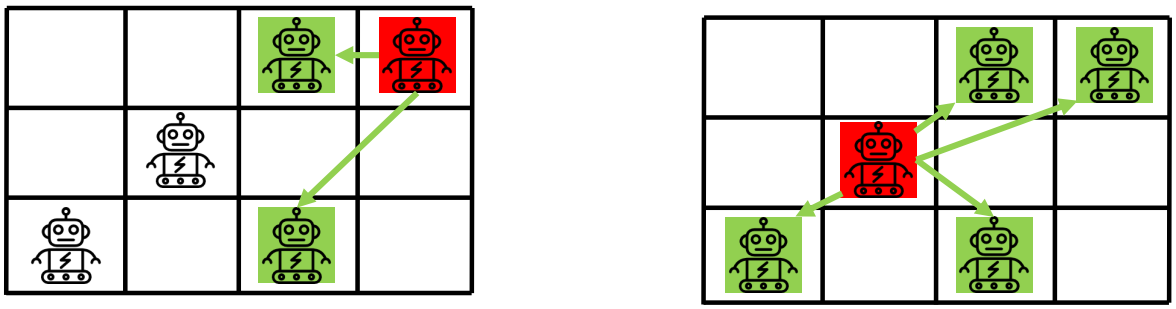


Figure 4.11: The world in a form of grid. Here, agents have a communication range of 2 cells. Both examples show the neighbours (in green) according to the red agent's point of view.

of message to communicate). Because all agents are cooperative, they will all increase (or decrease) at the same time, thus a variation the size of all agents in their neighbourhood is to be expected.

To counter this effect (and reduce variation), we integrate to the agent's behaviour a maximum improvement variation possible (called $Improvement\_Threshold$) and the capacity to evaluate its action impact on its neighbours' criticalities (see equation 4.17). To evaluate its action, an agent computes the impact of sending a message of volume 1 ($one\_message\_impact$, see equation 4.16 with $maximum\_crit_i$, the maximum value the criticality i can have), but according to its own $max\_process$ characteristic. To do this, an agent uses the adequate sub-criticality computation depending on the sub-goal, $crit1$ when the goal is to increase the communication, and $crit2$ when the goal is to decrease ($crit3$ is not used because the agent does not know if removing a specific message would lower the criticality or increase it). Thus, $one\_message\_impact$ is computed according to adding a communication volume of 1 after the $min_threshold$ is reached. From this computation, an agent can anticipate (partially) the impact of its decision.

The anticipation is done according to the agent's knowledge and environment. An agent wants to help another one to the point this latter's criticality is lowered by $Improvement\_Threshold\%$ (for example 5%). Then, it computes the impact it needs to have if all its neighbours do the same action as it does. This is done by computing the number of messages needed to lower the criticality and dividing it by the neighbourhood size, cf. equation 4.17. Note that neighbourhood size and the number of messages are both computed according to the communicating agent's characteristics and not from the agent that needs help. For example, an agent ag_1 has to decide the number of messages it sends, it has a $max\_process$ of 500 whereas its neighbour, ag_2 has a $max\_process$ of 400. Because ag_1 does not have access to the $max\_process$ of ag_2, it uses 500 as $max\_process$ to computes the $impact\_needed$ (so, it underestimates the impact of its message).

The $impact\_needed$ corresponds to the number of messages the agent needs to send to improve the criticality of the worst critical agent by $Improvement\_Threshold\%$ with regards to the size of the neighbourhood. For example, if ag_1 computes that ag_2 needs to receive 4 messages to improve its criticality by 5%, and that ag_1 has also ag_3 as neighbour, then $impact\_needed = 4/2$ (it increases its communication volume by 2). When $impact\_needed$ is not an integer, the agent sends a number of messages equal to $impact\_needed$ rounded down. Then it has a chance to send an additional message. The agent generates a random number (in the range $[0, 1]$) and if the generated value is lower than the $impact\_needed$ real part, then the agent increases its communication volume by 1 (summarised in the algorithm 4). For example, an agent computes the value 3.8 for $impact\_needed$, it increases its communication volume by 3 and then generates a 0.55 as number, in consequence, it adds 1 to its communication volume to send. $Improvement\_Threshold$ is a parameter given in the input of the system to increase/decrease the variation (and the speed). Furthermore, an agent does not decide to change the value of its variables if it would (from its perspective) increase the maximum

criticality in its neighbourhood. This behaviour can be problematic when the environment is dense. In this problem, the density is related to the difference between neighbourhood size and $max\_process$.

$$one\_message\_impact = \begin{cases} \frac{a_2 \times (min\_threshold_2 \times max\_process + 1)}{max\_process} \times 100 + b_2 & \text{if worst criticality } < 0 \\ maximum\_crit_1 - (\frac{a_1}{max\_process} \times 100) + b_1 & \text{if worst criticality } > 0 \end{cases} \tag{4.16}$$

$$\begin{aligned} nb\_messages\_needed &= \frac{Improvement\_Threshold}{one\_message\_impact} \\ impact\_needed &= \frac{nb\_messages\_needed}{nb\_neighbours} \end{aligned} \tag{4.17}$$

Algorithm 4: VariableToSend

Require: {each collections is ordered by utility}
$worst\_variables$: variables known as worse than those of the neighbour agents
$better\_variables$: variables known as better than those of the neighbour agents
$remaining\_variables$: variables without knowledge about their usefulness

1: **for all** variables with 1 as value **do**
2: $weight \leftarrow variable.weight$
3: **if** variable not in $worst\_variables$ and $cumul\_weight + weight <= max\_process$ **then**
4: $sendVariable(variable)$
5: $cumul\_weight \leftarrow cumul\_weight + weight$
6: **end if**
7: **end for**
8: **if** $cumul\_weight < max\_process$ **then**
9: **for all** variables in $better\_variables$ **do**
10: $weight \leftarrow variable.weight$
11: **if** $cumul\_weight + weight <= max\_process$ **then**
12: $sendVariable(variable)$
13: $cumul\_weight \leftarrow cumul\_weight + weight$
14: **end if**
15: **end for**
16: **end if**
17: **if** $cumul\_weight < max\_process$ **then**
18: **for all** variables in $remaining\_variables$ **do**
19: $weight \leftarrow variable.weight$
20: **if** $cumul\_weight + weight <= max\_process$ **then**
21: $sendVariable(variable)$
22: $cumul\_weight \leftarrow cumul\_weight + weight$
23: **end if**
24: **end for**
25: **end if**

4.3.7 Agent's Main Step Algorithm

The addressed problem is a DCOP problem. As such, many generic algorithms have been designed to solve these problems without concern about the application field. The proposed CODCOP algorithm (algorithm5)

is inspired by the Minimal Constraint Sharing Maximum Gain Sum (MCS-MGM) ADCOP algorithm. The two main differences between MCS-MGM and the CODCOP algorithm are:

- Instead of sharing constraints, the criticality is shared.
- Dynamic changes are taken into account.

At the initialisation, every variable is equal to 0 and then the loop began. First, they perceive their environment, see what variables are available and which are not. Then if there are criticality messages, they are treated. From the worst criticality it perceives, each agent decides the value of its variables. These values are sent to the neighbours (this is the information exchange inside the fleet of CAVs). From it, each agent evaluates its constraints to compute its criticality. Finally, it decides if it is worth it to share its criticality.

Algorithm 5: CODCOP

```
1:  Initialise all variables to 0
2:  while not terminated do
3:      Perceive environment modifications
4:      Receive neighbour's Criticality
5:      if most critical > 0 then
6:          Increase self-representation of neighbours' max process
7:      else
8:          Decrease self-representation of neighbours' max process
9:      end if
10:     VariableToSend()
11:     Receive neighbours' variables value
12:     Evaluate constraints
13:     Compute self criticality
14:     if |criticality| > |criticality| of the worst neighbour or SIGN(criticality) ! = SIGN(old_criticality) then
15:         send criticality to neighbours
16:     end if
17: end while
```

4.3.8 Cooperative Resolution

This section presents phenomena that can be observed during the agents' cooperative process. At the first step, no message is exchanged except the criticality one (which is the maximum positive value for each agent because all variables have 0 as value). When an agent receives its neighbours' criticalities, it increases the number of messages it sends according to algorithm 4. With the increasing amount of information exchanged, each agent improves its knowledge (which one of its information is useful to exchange). And, with the number of messages exchanged increasing, the global result increases as well. If an agent sees that the worst criticality in its neighbourhood is negative, then it reduces the number of messages sent. This process continues until no agent can modify its variables value without creating a new worst critical agent in its neighbourhood (according to its estimation).

Figure 4.12 illustrates the global resolution process in a simple way. First, agents synchronise together. Then they evaluate their constraint by computing the gain of their neighbour's communication (following equation 4.9). Then, they calculate their criticality according to equation 4.15. From this, they decide if they

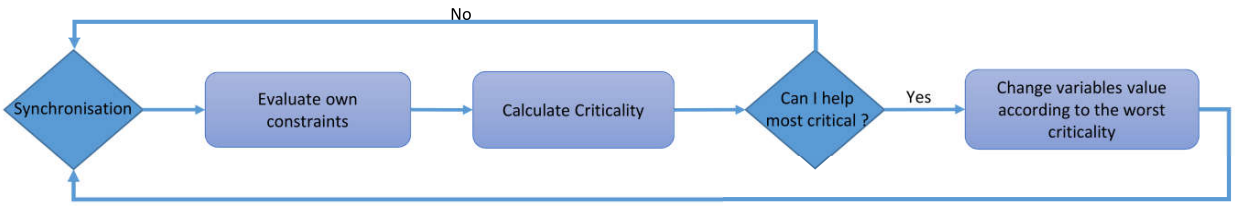


Figure 4.12: The five main steps of an agent's behaviour in the CODCOP module.

modify the values of their variables to help their most critical neighbour without creating a new even more critical.

4.3.9 Dynamic Adaptation

As stated earlier, the dynamic is an important part of the problems of the fleet of CAVs. Naturally, a system aiming at being functionally adequate can re-organise itself when confronted with changes. The problem can be modified in three ways:

- 1) an agent arrival/departure or modification (induced by the openness characteristic).
- 2) modification of a variable (appearance/disappearance or modification of its value).
- 3) constraints modification.

Next is described the system behaviour confronted to each disturbance. Note that these modifications are not exclusive and may happen in the same period.

The environment is not deterministic, meaning that CAV agents are not the only actors that may produce a change in the environment: any non-connected entity may modify it too. The weather is also a disturbing actor as it can change rapidly with rain and light. Consequently, as variables can appear and disappear, it has a direct impact on the agent's set of available variables. A non-available variable has its domain reduced to only 0 instead of $\{0, 1\}$. Old variables utility can also be changed and its perception modified. This represents either a modification of the object perceived by the sensors (i.e. the speed of an object), or the sensor can better perceive. A solution is to add a time component t to the variable tuple which defines the utility of the variable at a time t. Formally, a variable is defined by $x_{i,j,t}$ with i the environmental information, j the perceiving agent, and t the time when the variable takes the new utility value.

When the modification of the variable related information disturbs the system, two observations can be made on the system. The first observation is a loss in the global cost induced by variables put to 0 whereas their value was 1. This perturbation is rapidly compensated with agents' behaviour as they see an increase in neighbours' criticality. The second observation is a possible modification of which agent sends the information. With new variables available, the system enters once again in a re-organising state until it converges towards a new one.

The environment evolves when an ADP has made a decision and acted, as for example, when a new CAV agent enters the communication range of the fleet of CAVs (a new CAV connected to at least a CAV composing the fleet). Doing so, a new set of variables and values are integrated into the problem, and each neighbour agent sees its soft constraint modified with the addition of these new variables.

When an agent enters or leaves the system an increase of the maximum criticality can be observed because 1) all information from the agent that has left the fleet is no longer communicated and 2) the constraints of the

agent entering the fleet are different. The impact may be strong on the system, more on small neighbouring fleets because each agent has proportionally more impact on other agents criticalities. However, the remaining agents continue to send information they communicated before. That way, there is still a solution available that is improved with agents' behaviour.

It is also possible that the CAV changes its internal structure. Consequently to this action, the set of variables is modified and its soft constraint also is modified. This is represented by a modification of the cost of each variable inside the soft constraint. To model this, the time component t is added to the agent's soft constraint to differentiate it (cf. soft constraint equation 4.18).

$$cx_{i,j,t} = max([x_{i,0,t} \times utility(x_{i,0,t}), ..., x_{i,|A|,t} \times utility(x_{i,|A|,t})] \tag{4.18}$$

In addition to the soft constraint, resources may be limited in a specific context. For example, it is easier for an autonomous driving process to drive on a highway instead of in a city's dense traffic. To include this aspect, the time component t is also added to the hard constraint, specifically on the $max\_process$ which is susceptible to change.

$$ccom_{j,t} = \begin{cases} 0 \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l,t} \times w_{i,l} \leq max\_process_{j,t} \\ -\infty \text{ if } \sum_{l \neq j}^{|A|} \sum_{i}^{n} x_{i,l,t} \times w_{i,l} > max\_process_{j,t} \end{cases} \tag{4.19}$$

Constraints are strongly linked to agents' criticality computation. Criticality is calculated according to the $max\_process$ field. If its value changes, the agent criticality is also changed (exception if in both cases the $max\_process$ is exceeded). However, if the system was in a stable state (it has converged), the system enters in a re-organising state as soon as the first new modification of criticality is noticed. According to the agent's criticality new value, agents decide to switch some variables' value until a new convergence is reached.

Dyn-DCOP is the extension used to address dynamic problems. Dynamicity has a strong impact on an algorithm characteristic. Such an algorithm must be able to provide a solution before new modifications are encountered. The proposed solution based on the AMAS approach is dynamic compliant. Thus, by focusing on improving the most critical agent at every resolution step the realised assignment is incremental and the dynamicity easily taken into account. Moreover, the proposed agent algorithm is light and meant to be scalable.

4.3.10 Conclusion on the CODCOP Module

Constraint Optimisation is a domain that aims at solving a constrained problem while maximising (or minimising) an objective function. The information exchange inside a fleet of CAVs can be represented as a distributed optimisation problem with constraints.

Based on the well-known framework DCOP, a formalisation of the problem has been given, which uses two extensions of the DCOP framework: asymmetric and dynamic. Coming from these two extensions, the proposed solution addresses properties of **privacy**, **dynamic** and **openness**. The resulting DCOP formalisation shows that the research space grows quickly with the increase on the number of CAVs and environmental information, thus a local search method is to be preferred. Moreover, negotiation must be limited as such communication is added to the total volume of communications.

The solution proposed is based on the AMAS approach motivated by the distribution, dynamicity and local view characteristic inherent to a fleet of CAVs. Moreover, this approach is efficient for both challenges scalability and openness. Thanks to the bottom-up design, the local decision and privacy are respected. The proposed AMAS is a representation of the communication network of a fleet of CAVs, taking into account the communication range and mobility of the application domain. To respect the privacy of internal information, agents express their needs using the criticality concept representing the distance to reach their local objective.

To validate the proposed AMAS, many experiments have been conducted and are presented in the next chapter. The validation follows two steps: the first is the comparison with state-of-the-art algorithm; the second exploration on several characteristics is conducted to assess the efficiency.

4.4 Synthesis on the Contribution

Communication inside a fleet of CAVs is a powerful tool to enable cooperative strategy. Many cooperative strategies have been studied, designed, and developed in the last decades. The benefits of the cooperative strategy are no longer to be proven. However, cooperative strategies rely on the network's robustness and the social skills of entities composing the system. Most of the works make two strong assumptions: communications are fast, reliable and CAVs understand each other. The contribution detailed in this chapter addresses two challenges to ensure that these assumptions are true (communication is reliable and understood).

The proposition of this thesis is to use a Multi-Agent System formalisation to consider a CAV as an artificial unit while keeping the autonomous characteristic. A CAV is an agent evolving in an environment with other agents CAV and non-connected entities with no superior or central controller making decisions for the CAVs. The AMAS approach highlights two specific situations. The first one illustrates communication that is not understandable by the receiver. Instead of improving the overall system, such communication is useless and even can bring damages. The second situation is related to the limitation of the resources. Too many communication, especially useless ones, may prevent a CAV agent to take into account a useful one.

The contribution of this thesis is the design of two modules based on AMAS. The first module, called LUDA, enables an agent to transform a useless information into an understandable one. The LUDA module is an AMAS composed of three types of agents.

- *Data agents* that aim to find other *data agents* related to the same environmental information.
- *Morph agents* that transform the value of a *data agent* into the value of another *data agent*.
- *Group agents* that gather *data agents* representing the same environmental information and compute a better value for this environmental information.

The AMAS self-organises until it converges into a state that is able to transform information from another referential state into the agent's one. The module can continue to evolve through the CAV agent life duration.

The second module called CODCOP is used to optimise communications among the fleet of CAVs. CAVs are limited by their computation capacity and their necessity to decide an action in a short period of time. As a consequence, each CAVs seek to communicate the most useful information to others while avoiding the overloading of a CAV in the fleet. Using the DCOP framework and both asymmetric and dynamic extension the module respects the natural distribution of the control. Using local cooperation between agents, the module aims at optimising while addressing characteristics of information **privacy**, **scalability**, **openness** and **dynamic**.

The validation of the contribution is presented in the next chapter by comparing the result with state-of-the-art methods that can be used for the same purpose, as described in chapter 3. Then, each module is confronted to the challenges of **dynamic**, **openness** and **scalability** to evaluate their efficiency towards them.

Chapter 5

Experiments

Objectives of this chapter:

- To give the experimental conditions
- To evaluate the LUDA module with regard to the challenges of dynamicity and scalability
- To observe how the CODCOP system behaves with regard to disturbance and evaluate its adequacy with time constraint
- To conclude on the strengths and weaknesses of the contribution

---

This chapter presents the experiments realised to evaluate the LUDA and CODCOP modules presented in the previous chapter. Experiments are conducted in two steps: first, each module is compared to state-of-the-art methods to give an insight on strengths and weaknesses. Then, evaluations are done to validate corresponding challenges.

To ensure that experiments are not biased because of the hardware configuration, experiments presented in this chapter are all realised on the same computer. Its characteristics are:

- A Windows 10 Professional operating system.
- An Intel Core I7 8th generation for the processor.
- All data are stored on the same SSD, locally accessible.
- No graphical card.
- 32 GB of random access memory.

This chapter is constructed using the following outline. First, the definition and the construction of a synthetic environment using an AMAS approach are given. Then, the experimental conditions are explained and justified. The next section aims at evaluating the LUDA module evaluation according to the two steps validation process. This module is compared to data estimation state-of-the-art methods. Then, the communication optimisation experiments on the CODCOP system are presented, using the same process as the LUDA one. Finally, a synthesis of both modules is given.

5.1 Synthetic Environments Generator

One of the means to validate a system is the experimentation process, commonly used when no theoretical proof is available. Through many scenarios, hopefully covering all the possible spectrum, the provided results have to validate the efficiency on one or several criteria. However, the experimentation process requires data that are not always available. A solution is to generate artificial data to replace the absence of real data.

In this thesis, the Autonomous Driving Process (ADP) has been considered as a black box taking several inputs and giving one or several outputs. So, to evaluate the LUDA module, it is required to have a black box simulating the ADP. As a level 5 ADP is not currently available to conduct experiments, we propose to generate several black boxes that simulate a complex decision system. The inside of the black box has to be impossible to analyse. The synthetic black box generator developed is inspired by the system BACH designed by Boes [18].

5.1.1 Black Box Generator Requirements

Usually, control systems are first tested on simulations, before being applied to the real process. These simulations are based on accurate models of the target process and sometimes can be very difficult to obtain. However, in the case of a black-box approach, the fidelity of the simulation to the real system is not required. In other words, the model used by the simulation does not need to reproduce precisely the behaviour of a particular real system, or even to have its semantics. For example, a micro lane merging simulation has to reproduce the vehicles dynamics. An abstract black box has to reproduce a **type** of behaviour, the one of the real system. Where a simulation has to mimic the real situation, an abstract black box only has to mimic the complexity. From the lane merging example, an abstract black box reproduces the **complexity** of the vehicles computation. If the computation requires three successive functions, the abstract black box also uses three successive functions. However, the functions used by the abstract black box can be any mathematical function.

A large part of the difficulty of using an abstract black box to experiment a system is linked to the adaptation. Experiments concern the system's ability to learn and adapt to a dynamic environment. As the system has no knowledge of the internal working of the system it is linked to, it does not need to know the semantics to control it. Thus, it is possible to use inputs and outputs that do not correspond to anything concrete to experiment the process.

Complexity can take three forms for an abstract black box. The first one is the interdependence between inputs and outputs. An input may influence one or several outputs. How an input influences an output can be susceptible to the input value (for example with the mathematical function SIGN, depending on the input sign, return -1, 0 or 1). Second, an influence does not always follow a linear scheme with accelerations, variations, and amortisations. Last, noise adds another dimension to the complexity because it does not follow the usual behaviour.

Experimenting with a system that needs to adapt several configurations given by the environment requires to observe it under many different configurations. The use of an abstract black box enables the experimentation on a specific configuration and many variations from it to observe the concrete system behaviour. Because the black box variation results from the modification of input-output interdependence, it is possible to create some mathematical trends enabling us to observe the system confronted to these specifics trends.

With the design of such an abstract black box, it was possible to experiment with the LUDA module without a real ADP.

Abstract Black Box

An abstract black box is constituted of three types of components: **inputs**, **outputs**, and many **mathematical functions**. The number of inputs or outputs is not limited, and only inputs and outputs are visible from an exterior point of view. Each input is linked to at least one output through a path including several functions and reciprocally. A function has at least one input that can be an abstract black box's input or the result from a function (itself included); this means, the resulting black box may include a cycle between the input and the output of the same function. An output has for input a result from a function (to avoid the output to use the non-modified value of an input). An example with 4 inputs, 7 functions, and 3 outputs is given in figure 5.1. The internal arrow express a link between two components of the abstract black box where the source one gives a value to the destination one. For example, the function $F1$ requires the value from Input1 and Input2. In figure 5.1, to obtain the *Output 1's* value, the following steps are made: 1) the value of *Input 1* is used as an input to *F1* and *F2* and the value of *Input 2* is used as an input to *F2*; 2) the values of *F1* and *F2* are used as input to *F4*; 3) the value of *F4* is used as the input to *F6*; 4) the value of *F6* is the value of *Output 1*. Here an input can be used by more than one function. Indeed, if the abstract black box represents an ADP, inputs are environmental information observed thanks to the CAV's sensors. According to the speed of the car just in front, the ADP decides the best speed to adopt. But the car's speed can also be used by the same driving process to decide the best trajectory. So, one input can be used for two outputs.

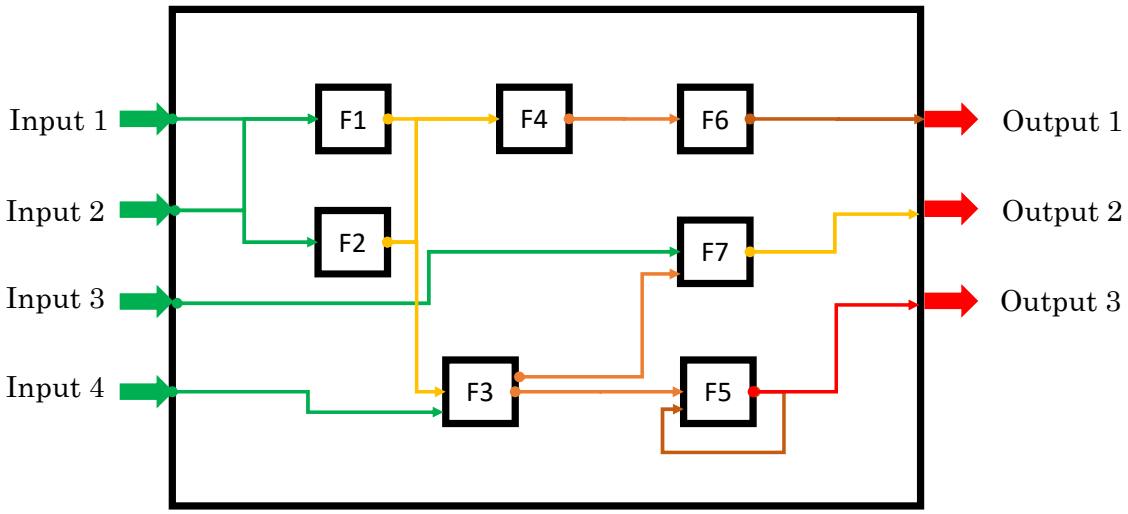


Figure 5.1: Example of an abstract black box

User's Requirements

A generator of an abstract black box aims at being used for many applications. So it has to be adapted to its problem specificities. A user must be able to specify:

- The number of the black box's inputs.
- The number of the black box's outputs.
- The number of the black box's functions. This number is optional to leave this decision to the black box generator. Doing so, the abstract black box generates a number of functions according to the outputs and inputs constraints (given by the user).

- The functions' minimal number of inputs, which is by default 1.
- The interdependences between inputs and outputs that define which input influences which output.
- The required complexity degree of the outputs, given by the user. In the previous example, an output may be the result of only one function or several. In the example, output 1 is the result of 3 successive functions while output 2 is the result of two successive functions. Output 1 has a complexity degree of 3 and output 3 has a complexity degree of at least 2 (or more because of the cycle).

The AMAS approach has been chosen to design an abstract black box generator according to user requirements. This is motivated because an abstract black box can be simulated by a complex system (too complex to understand its internal working). Consequently, inputs, outputs, and functions are agents and the overall system is a self-organising system. The use of a self-organising system to obtain an abstract black box prevents a deterministic construction of the black box.

Agents

The black box generator is composed of three types of agents: *input agent*, *output agent*, and *function agent*. An *input agent* only requires to be linked to a minimum number of functions inside the black box, thus its objective is to reach the minimum number of links. This number is given by the user. An *input agent* perceives every *function agent* in the black box. An *input agent* can create a connection between it and a *function agent*'s input port. To be satisfied it seeks an available *function agent* and creates a connection with it. It repeats these actions until the minimum number linked is reached.

An *output agent* has a first constraint that ensures that it is linked to one and only one function's output port. The constraint is related to the **complexity degree** specified by the user. The *output agent's* second constraint is to only be linked to a function whose output has a matching complexity degree. Its objective is to satisfy these two constraints. To do this, it creates a link with an output port of a function.

A *function agent* has the objective of having both its ports (input and output) linked to at least another agent in the environment. This agent is created with a mathematical function (specified or randomly generated), that uses the input(s) value(s) to compute a result. When the number of inputs exceed the mathematical function number of inputs, inputs values are aggregated until the remaining number of value match the minimum number of inputs. For example, the mathematical function used is the product of two numbers. With three inputs, two of these are aggregated, and then the product is made. A *function agent* can create a connection between its output and a *function agent*'s input or an *output agent*. However, this action is only available if the function has its input connected to at least one other agent. A *function agent*'s input is characterised by a complexity degree. The complexity degree is equal to the highest complexity degree in the path used to reach the function. An input coming from an *input agent* has a complexity degree equal to 0. For each successive function in the path, the complexity degree increases. In figure 5.1 the input's complexity degree of F3 is 1 and the output is 2.

Cooperative Process

To obtain an accurate black box, agents organise themselves to satisfy every agent in the system. Concretely, agents create connections between them to satisfy every constraint. At first, no connection exists, and only *input agents* can act. Each *input agent* creates a link with a function according to its characteristics (number of inputs). Functions that have their input constraints satisfied can create a connection with other functions or output. Once every agent is satisfied, the system has converged into a stable state and the abstract black box is

functional. Note that agents prefer to limit their output connection to only one but can increase it if necessary, as seen with F2 in the example in figure 5.1.

However, it is possible that agents cannot be satisfied when the user requirements are overconstrained. Agents are faced to a Non-Cooperative Situation (NCS) of incompetence (there is no action available). For example, no function has achieved a complexity degree enough to be used by an *output agent*. This NCS can be solved in three ways:

- A *function agent* can decide to remove a connection with a low degree of complexity. Doing so, it adds a complexity constraint equal to the *output agent* degree - 1.
- A new *function agent* is created with a complexity constraint equal to the *output agent* degree - 1 on its input port. This action is preferred when there is no user requirement on the number of functions inside the black box. Indeed, in this case, adding a new *agent function* has no conflicting effect with any constraints in the system.
- A *function agent* creates a connection between its output and one of its input ports (see $F5$ in figure 5.1). Doing so, it can increase the complexity degree to the infinite.

To decide between the first and the third action, agents have a criticality value according to their number of outputs connections and the number of cycles, with the objective to homogenise the number of connections between each agent.

Another incompetence NCS that may happen is the lack of agents available to create a connection. As an illustration, suppose all functions have a complexity degree constraint on their input port or several inputs constraints. In figure 5.1, the *input agent* 3 requires a connection, but all *function agents* are already satisfied. An agent that needs to connect its output port cannot do so. Solutions available to solve this situation are:

- To create a new *agent function*. However, this solution may have two drawbacks: 1) the new function can be in a non-cooperative state if it cannot connect to anyone after and 2) it may violate the user's constraint on the number of functions.
- Another *agent function* can increase its number of inputs to help it. In consequence, a new connection is available. This choice is made according to agents characteristics and their possibility to help.

The last incompetence NCS is related to the input-output inter-dependency. Not enough inputs are available for the inputs of a *function agent*. Note that in the case of inter-dependency constraint function, this constraint cannot be violated for any reason. To solve this NCS, two actions are available:

- To create a new *agent function*. Note that this choice can be dependent on user requirements and the required complexity degree. If the number of functions required is fixed, then no new *agent function* can be created.
- The *function agent* reduces the number of required input connections by one, with a minimum of one.

The organisation process continues until a stable configuration is achieved. It is possible to dynamically change the black box, in that case, agents will self-organise once again.

Functioning Illustration

At the beginning, the user enters its requirements that become the problem constraints. From these, agents are created and start the cooperative process. Connections between agents are removed and created until every agent is satisfied. Once agents are organised one with each other, the system is ready to be used. The user gives a value for each input of the abstract black box and gets a value for each output. According to the black box illustrated in figure 5.1, figure 5.2 gives an example of use once the black box is organised. The user has given the values 2, 6, -3 and 1 for inputs, and it receives in outputs the values 3, 0 and 12.

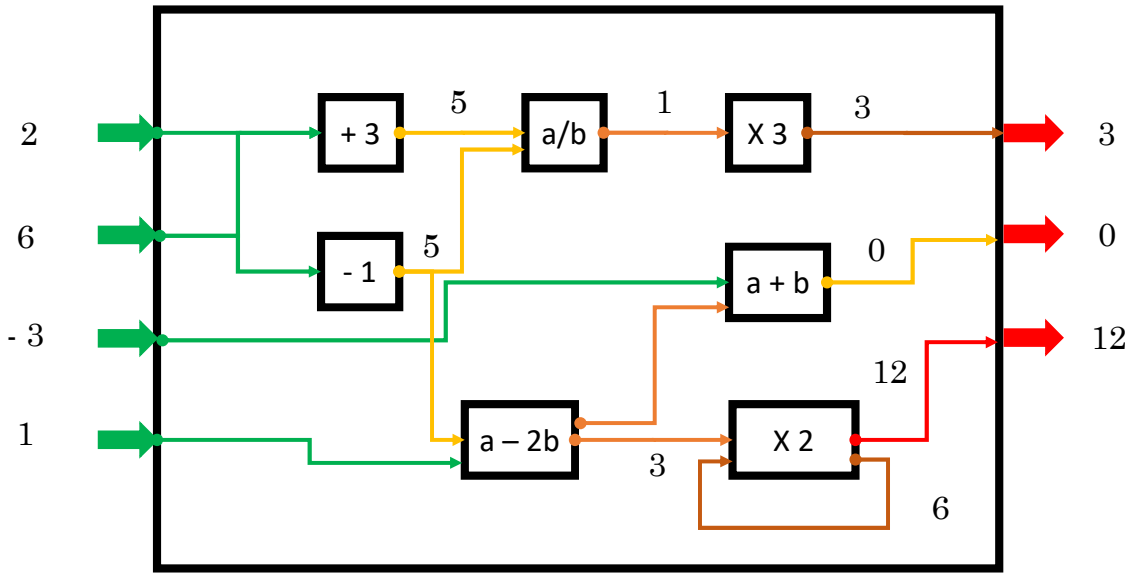


Figure 5.2: Example of an abstract black box usage with 2, 6, -3 and 1 as inputs.

5.1.2 Synthetic Dataset

The validation of the data estimation method should normally rely on the use of a dataset extracted from real sensors. These datasets are often divided into two sub-datasets: the first one used to train the system and the second used to evaluate the system. However, datasets are often limited and/or impossible to access. Finding the accurate dataset to evaluate certain behaviours can be time-consuming and/or costly. An alternative for the user is to create a dataset from scratch with the properties he is looking for. This dataset is called a *synthetic dataset*.

With the use of a specific abstract black box presenting only one output, it is possible to generate values for one data, so allowing to generate synthetic data. Two solutions are possible: 1) create as many abstract black boxes as the number of data wanted by the user, or, 2) create one black box with as many outputs as the user wants.

5.1.3 Synthetic Noise

The quality of collected real data depends on environmental conditions and sensors quality. So validation has to observe a system behaviour confronted to data errors or inaccuracy, in this case, the data is called *noisy*.

Among the differents forms that noise may take, the most common noise is the Gaussian one [13]. Gaussian noise is a statistical noise that ensures a Gaussian distribution of the noise. It is characterised by a

Probability Density Function (PDF) equal to the normal distribution. The probability density function p of a variable v is given by equation 5.1, where μ is the mean of v and σ is the standard deviation.

$$p(v) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(v-\mu)^2}{2\sigma^2}} \tag{5.1}$$

This kind of noise is caused by a calibration error or by the hardware itself. This is why some sensors are more likely to **overestimate** a value while others **underestimate** it. In that case, the noise must follow this trend. Gaussian noise can also be used by shifting the centre of the Gaussian curve.

It is possible to add noise directly inside an abstract black box by adding a specific function just before an output (in case of different noises, it is possible to add more than one). This function is different from the others and is not considered as an agent as it does not participate in the overall organisation. This function needs to be activated by the end-user whenever he desires to add noise. The corresponding mathematical function is by default a Gaussian one but another can be specified by the user. In further sections, Gaussian noise is used by default.

5.1.4 Synthesis on Synthetic Environment

This section has presented the tools used to evaluate the LUDA module. Indeed, such an evaluation requires an ADP to obtain feedback. As it was impossible to have such an ADP, we propose (i) to consider it as a black box with inputs and outputs linked together by a chain of mathematical functions and (ii) to generate such an abstract black box supposed to simulate its functioning. This generator based on the AMAS approach, enables to obtain an organisation ensuring enough complexity that prevents an easy comprehension of the inside.

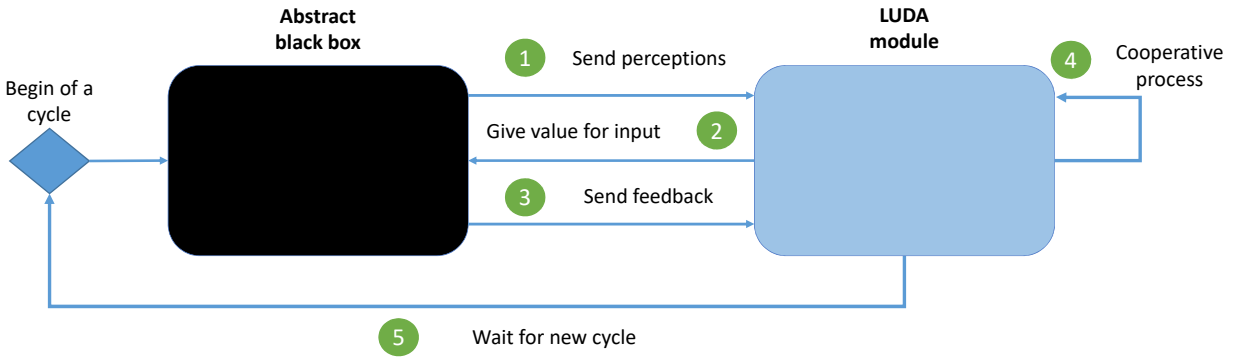


Figure 5.3: Global Process used to evaluate the LUDA module

5.2 LUDA Module Experiments

The first contribution of this thesis, the LUDA module gives tools for an agent to modify a received information to understand it. This is similar to the a value estimation that data estimation state-of-the-art methods provide.

To compare the LUDA module to these methods, we use an abstract black box in place of the Autonomous Driving Process (ADP). Figure 5.3 shows the experimental process used for the LUDA module. This is motivated because no level 5 ADP exists/is available yet. To be usable, the abstract black box purpose is to simulate the **complexity** of the ADP decision (given the input, it is nearly impossible to predict the ADP output). The inputs of the black box represent the CAV perceptions, and the outputs the CAV actions. ”Perceived” data are synthetically generated and sent to the LUDA module ❶. Then, the black box requires from the module a value for each one of its inputs. From these values, one value corresponding to the output is given ❷. This value is then compared to the value that the abstract black box would have given with the perfect values, it is the oracle truth. In this context, the oracle knows the correct value to give as input to the abstract black box. Then the abstract black box sends to the module the oracle values ❸. The LUDA module is activated and self-organised (activation of all agents in the LUDA module) between each cycle of the abstract black box ❹. This cycle consists in computing the value for each output and then modifying the value of each information. And a new cycle begins once again ❺.

A second way to evaluate the LUDA module is to use an already existing dataset with state-of-the-art methods and to compare the results of each one of them with the results of the LUDA module on the same dataset.

To do this, first the experimental methods and metrics used are presented. These two types of evaluations are then shown. The next section is devoted to the study of the adequacy of LUDA regarding scalability. Finally, a conclusion examines the LUDA module with regard to the challenges of **dynamic**, **openness**, and **scalability**.

5.2.1 Experimental Methods and Metrics

The correct evaluation of any new system/method/algorithm needs to answer the questions: ”does it provide something that no other technique does (in the large meaning) ?” and ”what are the strengths and weaknesses ?”. The first question is answered by comparing obtained results with those of existing techniques. The second requires the use of accurate metrics to evaluate the system on a specific aspect. For example, dynamicity can use the computation time metric.

Three standards techniques are used to compare the LUDA module results. The first technique used is

a simple linear regression algorithm, the second is the use of a random forest algorithm and the last is a gradient boosting algorithm. They have been chosen for their characteristics because they are not time and computational consuming. When dynamic is not in the model particularity, a new model is computed when required. Neural network and extensions are missing, because of two main reasons: 1) it is computational and resources expensive, so not adapted with the objectives and 2) the lack of an oracle stating which information correspond to which other in a different referential frame is a major issue. The LUDA module has been developed in Java and uses no specific library. Some libraries have been used to observe the inside of the module.

Three state-of-the-art techniques are borrowed from the scikit learn library [113]. To evaluate that the LUDA module results are correct and to reduce experiments time, only useful data are transformed. According to the LUDA formalisation, these techniques only learn the information in another referential frame information (e_d seen in section 4.2.1 of chapter 4) transformation with the matching information in the agent's referential frame (p_d). The part of dealing with which information is related to which other is not taken into account for the state-of-the-art methods. This is motivated by the fact that only final results are interesting for the comparison.

Comparisons are realised with a similar dataset for all techniques, LUDA included. The common method is to give a part of the dataset for the training (80%) and the remaining part is used to evaluate the training efficiency. To follow the fleet of CAVs application case, before getting the true value of the corresponding information, a prediction is made about the missing information.

After validating the LUDA module as effective in a data estimation way, there is a need to validate its efficiency for the application case. As stated earlier, the environment is dynamic, open, scalable and subject to errors. So the efficiency evaluation focuses on:

- 1) **The time** required to achieve an organisation cycle. Dynamic induces that changes may occur at any time. If more time is necessary to achieve one organisation cycle, there is more chance that modification occurs in the environment. When the system is adapting to this environmental change, it is not in a functional state to give a value when required.
- 2) **The number of examples in the history** required in memory to achieve a correct state. Perceptions are acquired continuously and it would take a few times before the volume increased to a point that it becomes an obstacle to the smooth operation.
- 3) The **number of necessary cycles** to obtain an accurate organisation, also known as convergence time.
- 4) The **size** of the E_D (communicated information) and P_D (perceived information) sets. The bigger, the more calculations. This metric enables to assess the scalability.
- 5) **The noise** that alters the dataset used by the LUDA module. These experiments are conducted with two datasets. One with the true value for each data and an other one with a noisy value. Training is done with the noisy dataset.
- 6) **The resulting group organisation**. It enables to know the number of *data agents* that are correctly placed, the number of *data agents* wrongly placed and the number of completed groups. A group is considered as complete if there is no *data agent* outside this *group agent* that its information is related to one of its inside *data agents*.
- 7) **The number of remaining agents** once the system has converged. The number of *Morph agents* is proportional to the size of the dataset. Their possible self-destruction action prevents them from being too numerous, reducing the total number of *morph agents*.

Metrics 1 to 5 are used to evaluate the functional adequacy of the LUDA module toward the problem of **cross-understanding**. Metrics 6 and 7 are used to evaluate the agent's organisation inside the LUDA module.

Datasets

Many datasets and abstract black boxes have been tested, many offering the same conclusion on results. The selected dataset shows the evaluation of the previously described metrics. These dataset structures are summarised in the table 5.1. A noise categorised as "Estimation" means that the noise is not centred on the real value but is offset from the real value, to create over-estimation and under-estimation according to sensors. The noise can be distributed over "5%" for every data or being randomly distributed between 0 to 10%; in this latter case, it is called "Diff". Two label types are used in synthetic datasets according to the information source. If the information is from P_D (the CAV perceives the information) then the label follows this semantic: "VType"$< number >$. Otherwise (the CAV receives the information), the word "copy" is added at the end: "VType"$< number >$"copy"$< sourcenumber >$. For example VType1 and the transformed copy VType1Copy:0.

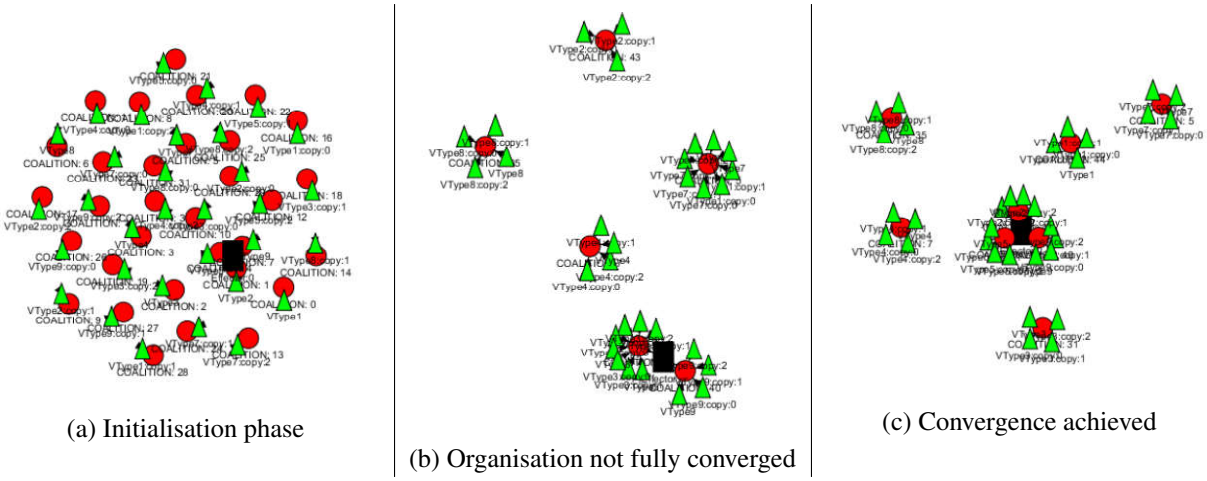
Dataset Name	Generation	Size of P_D	Total number of data agents	Noise	Experiment Objective
Emiliana	Real	5	20	No	Real data observation
D68N	Synthetic	17	68	No	Synthetic data baseline
D68N5	Synthetic	17	68	Gaussian 5%	Noise influence
D68N5O	Synthetic	17	68	Estimation 5%	Noise influence
D192NO	Synthetic	48	192	Estimation Diff	Scalability
D392NO	Synthetic	98	392	Estimation Diff	Scalability

Table 5.1: Characteristics of datasets used to evaluate the LUDA module.

5.2.2 LUDA Module Organisation

In this subsection, some results about internal organisations are presented. For this experiment, the dataset D68N5 is used to have a better understanding of the system state. A bigger dataset with many entities gives a resulting organisation that would be difficult to read. The organisation shown in figure 5.4 is a snapshot extracted from the real execution of the LUDA module using a java library called "Links". This library allows to save snapshots of a system. In the LUDA module case, a snapshot is taken at each end of an organisation cycle. A snapshot includes every *data agents* (green triangle), *group agents* (red circle), and the black box (black rectangle). *Morph agents* are too numerous to have a readable image. *Data agents* and *group agents* can be linked by an arrow signifying a grouped relation between them. In the following examples, only 3 *group agents* are linked to the black box, for visualisation purposes. Meaning that only 3 p_{D_i} are used by the abstract black box of these examples. This is not an issue as the objective is to observe groups and it lightens the image.

Figure 5.4 provides an example of a correct organisation with a full convergence. All information is available at the first cycle. One *data agent* is created for each information. At the initial state, each *data agent* is the only one linked to its *group agent*; so there are as many *group agents* as *data agents*. The experiment starts and agents begin to organise themselves. As seen in figure 5.4a, in the early cycles no organisation is obtained making the module not able to provide a correct value if required. But then after several cycles, the group number decreases thanks to the *data agents*' grouping. Convergence is not fully achieved but every group includes more than one *data agent* as seen in figure 5.4b. In consequence, in the



(a) Initialisation phase

(b) Organisation not fully converged

(c) Convergence achieved

Figure 5.4: LUDA module organisation confronted with the D68N5 dataset.

case of missing information from P_D, the LUDA module can provide a value thanks to communication. After a few additional cycles, *data agents* manage to get in the right group and the system has converged, see figure 5.4c. Finally, each *group agent* includes 4 *data agents* as it should be for this example, due to the dataset used. The convergence is achieved and the resulting organisation matches what was expected. Convergence means that if an information is perceived by one agent in the neighbourhood and communicated by it, the LUDA module can provide a value that is usable by the ADP.

5.2.3 LUDA and Data Estimation

This section aims at giving a comparison between the LUDA module value prediction with state-of-the-art ones. In the LUDA module, *morph agents* estimate their objective data value using a linear function they have adapted through multiple processes. This is mainly used to group related *data agents*. Then, a *group agent* uses this value with a specific weight for its pondered sum and computes the value to send to the black box. If all agents have this cooperative behaviour, a *group agent* can send the most accurate value according to its skills.

The following experiments show aggregated results of every dataset presented in table 5.1. For each dataset, 100 lines (with a value for every data) are taken into account (thus the number of cycles is 100). After each cycle, a prediction is made for each information. State-of-the-art techniques re-build their models after each cycle.

Figure 5.5 shows a transformation function that a *morph agent* has constructed from its history (in blue) and (in red) the objective value (here called Oracle). This example shows the transformation function between two related non-noisy data. The peak at cycle 77 shows a bad prediction but the agent modifies quickly its function according to it. This experiment shows how a *morph agent* transformation function allows the agent to estimate the objective value.

The second figure 5.6 regroups all ”good” *morph agents* results and compares these to state-of-the-art methods for all datasets. The error is the absolute value between the real and the predicted value. The value is included between 0 and 100 for the illustration. *Morph agents* are more subject to anomalies than other state-of-the-art methods. However, the average error is kept low and is better on average. According to this figure, there is a need to reduce the anomalies impact which is the role of *group agents*. All four methods have many points outside the box because the prediction is made while organisation/learning is not over. Another

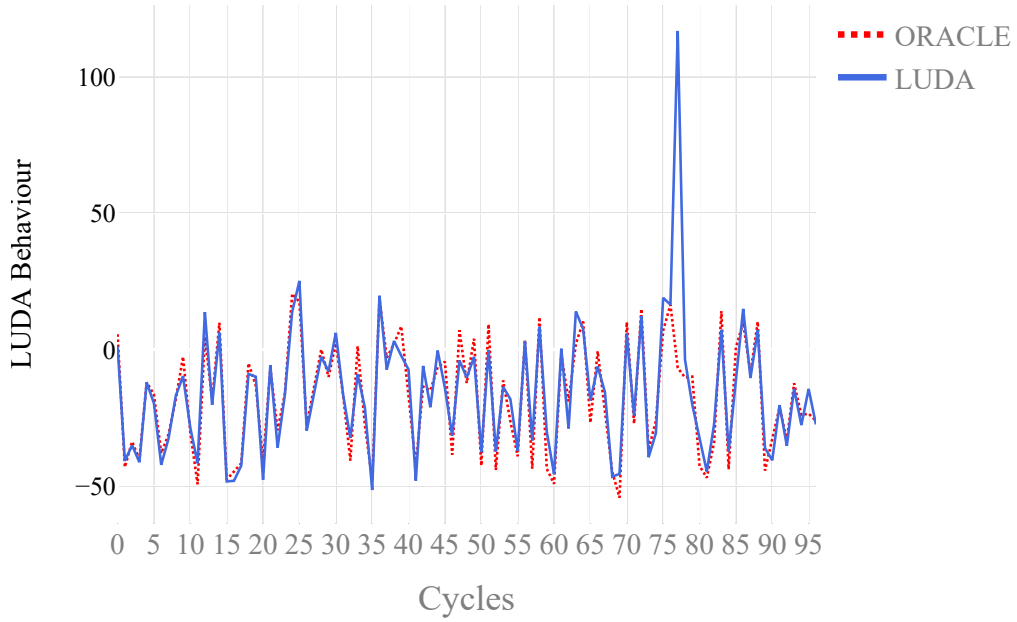


Figure 5.5: The result of a morph agent adapted linear function. The red line, oracle, is the true value, and the blue one is the value given by the agent.

cause is the noise disrupting the overall prediction.

Both figures 5.5 and 5.6 shows that a *morph agent* transformation function behaves such as the standard techniques of *linear regression*, *random forest*, and *gradient boosting*. However, individually these techniques are subject to anomalies and noise. *Group agents* aim at improving the accuracy of the value sent to the ADP. In the following section, the *group agents* efficiency is evaluated.

5.2.4 Characteristics Exploration of the LUDA Module

This section shows the *group agents'* results using previously described metrics of **group organisation** and **remaining agents**. Then, an evaluation of the *group agent's* error prediction is made. After, the characteristic of scalability is evaluated with the two metrics of **time** and **number of agents** according to the size of E_D and P_D. An exploration of the number of historic required for *morph agents* is examined before concluding on the LUDA module.

Resulting Group Organisation

The LUDA module functions adequately when achieving a complete organisation of all its parts, even if, it has no access to the organisation objective. This emerges from the agents' behaviour. Depending on the data nature, *data agents* can be wrong about their group. This may happen when different information are close enough to cause any ambiguity. With the increase of information number in the environment, the likelihood of this happening should increase.

The first study is related to **scalability** and **noisy** metrics. It aims at understanding how the LUDA module behaves faced with a more and more complex environment. To do this, the module is evaluated with all datasets and the resulting organisation is evaluated at cycles 30, 90, 150, and 180. Two metrics are used to evaluate it. The first metrics is the percentage of Completed Groups (CG). A CG is a group including one p_{d_s}

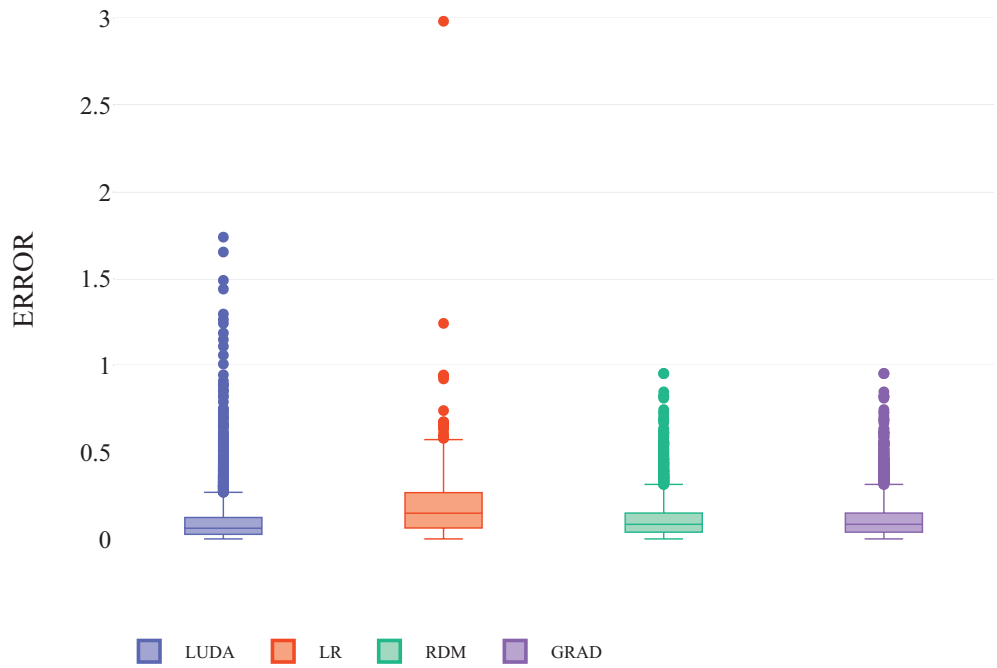


Figure 5.6: Difference between the true data and the value estimated with LUDA module adaptation, Linear Regression (LR), RanDom Forest (RDM) and Gradient Boosting Regression(GRAD).

(a perceived information) and all related e_{d_j} (all received information that represent the same environmental information in another referential frame). If one *data agent* is missing or in another *group agent*, it is not considered as completed. In other words, a CG is the final organisation objective of all agents composing it, even if they are not able to know it. The second is called Well-Placed Data (WPD). Less strict than the previous one, a *data agent* is considered as WPD if it is in the same group as p_{d_s}. However, this knowledge is never accessible for any agent, so a *data agent* does not know if its group is the correct one.

Table 5.2 shows the resulting groups at different resolution cycles. The first observation that can be made is the 100% accuracy in a non-noisy environment. Other datasets are not shown because results are identical. When noise is added, the LUDA module does not converge immediately. In an environment with no noise or with similar noise distribution, group accuracy stays high. In the case of different noise values, the noisiest observations are stored in memory to be used for the adaptation. Noise degrades the efficiency of the adequacy computation, leading to less efficient agent interactions and fusions. Moreover, the increasing number of data in the environment improves the results, accuracy increases from 85 to 93 for WPD at the 180^{th} and the number of CG increased from 88 to 94. The LUDA module is then more efficient in a large scale noisy environment than in a low-scale. In consequence, the LUDA module is good for **scalability**.

Cycle	30		90		150		180	
Dataset	CG	WPD	CG	WPD	CG	WPD	CG	WPD
68 data No noise	100	100	100	100	100	100	100	100
68 data 5% noise	88.24	92.65	88.26	95.59	94.11	95.59	94.11	98.53
68 data Diff noise	70,59	85,29	88,24	97,06	64,71	88,24	64,71	85,29
192 data Diff noise	75,51	90,10	73,47	94,27	73,47	91,15	71,43	90,63
392 data Diff noise	73,74	89,54	75,76	92,86	76,77	92,09	78,79	93,62

Table 5.2: Formed Groups results according to the percentage of Completed Groups (CG) and Well-Placed Data (WPD).

Group Agents Prediction Evaluation

This section explores the advantages of using several data using the LUDA system in a noisy environment. This exploration is correlated to the comparison section because realised experiments are similar; nothing is done with state-of-the-art methods. In this context, only LUDA's results are studied with different noise values and distributions. The objective is to observe how the LUDA's behaviour is impacted by the different noise values. Four scenarios are used for the experiment:

- 1) the use of a standard Gaussian noise distribution with 1%;
- 2) the use of a standard Gaussian noise distribution with 5%;
- 3) the use of a random strategy on the noise value used for each data. For example, data can use a Gaussian noise of 5% and another of 10%;
- 4) The simulation of over-and under-estimation trends. This is achieved by shifting the centre of the Gaussian noise from the real value. The noise is different for each data;

Figure 5.7 shows the value difference between the real data and the data adapted by a *group agent* following equation 4.1 in the chapter 4. As expected, the smaller the noise, the better the results. there is a small difference between middle-boxes whereas the noise can reach 10% in the latter case. The impact of the increasing noise seems to be compensated by the *group agents'* strategy. This is illustrated because the increase of error is small. The last box illustrates that when a *group agent* can compensate values, the error is reduced even with an even distribution of the noise value.

LUDA is more effective when different sources are under-estimated or over-estimated. There is a relation between the peaks observed in figure 5.5 that have a direct negative impact on the ADP result. Indeed, when the transformation function gives a value far away from the true value, the corresponding *group agent* uses it. But anomalies (one-time error) are still an issue. A LUDA improvement would be (i) to detect when the value of a *data agent* is far from those proposed by other *data agents* situated in the same *group agent* and (ii) to exclude it temporarily. This improvement is not easy to implement because there is no way to know the most accurate *data agent* in a *group agent* when a one-time error occurs. In consequence, excluding this *data agent* could have a worse impact than keeping it. This point needs further investigations.

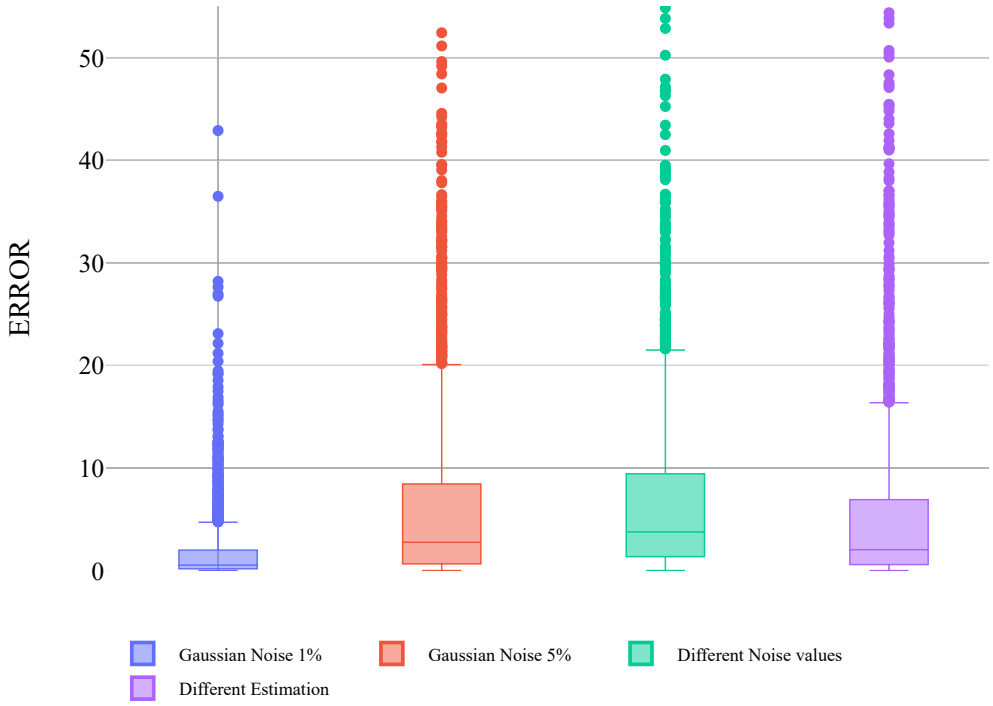


Figure 5.7: Effectiveness of LUDA on different noise values. The error is the sum of all differences in the experiment.

Exploration of Computation Time

To consider the computation time, experiments look at the number of remaining *morph agents* compared to the ideal number, as well as the time to achieve a resolution cycle. In the LUDA module , two steps can be particularly computational expensive. The first one concerns the *morph agent's* behaviour. The algorithm itself is not computational expensive as it iterates only over 20 examples. But the number of *morph agents* grows exponentially with the number of data. Parallelising every *morph agent* is not currently possible, so a study focusing on the *morph agent* cycle is done.

Figures 5.8 and 5.9a show impact of the data number on computation time. In the worst-case scenario, with an increase by a factor of 160 in the number of agents, the time increases by only 100. Note that in these experiments, no agent is parallelised as it was not possible to parallelise every *morph agents*. So to keep results not subject to this number, *morph agents* are activated sequentially. The algorithm complexity can be formulated with equation 5.2. The value 3 is the number of examples used to adapt a *morph agent's* transformation function, and is the only loop in the algorithm. So, each *morph agent* loops three times each cycle, and the maximum number of *morph agents* is equal to the square of the number of *data agents*. In the worst-case scenario, all data are available and each *data agent* creates a *morph agent* to link it with a *data agent*. Thus all the *morph agents* can adapt their function at the same time. During the adaptation of the coefficients of *morph agents*, the destruction of useless *morph agents* decreases the computation time as seen in the cycle 25 of figure 5.8. In the worst-case scenario, when the totality of data is available the computation time is high until the destruction of several *morph agents*. A peak appears at cycle 17 (after the creation of all *morph agents* but before *morph agents* begin to destroy themselves) and then the number of remaining *morph agents* reduces until the 35^{th} cycle from which it is stabilised. In a real case, with a computation time constraint, the number of data to process in one cycle should be limited. A considered strategy consists in

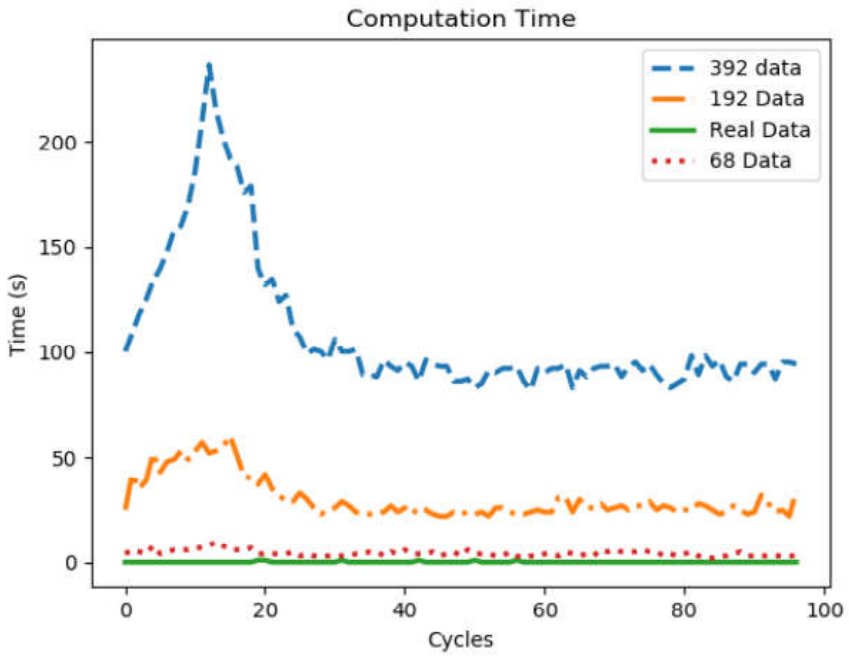


Figure 5.8: Evolution of the *morph agents* coefficients adaptation

selecting only the most critical *data agents* and in improving the *morph agents* adaptation. Improvements must be done on the *morph agents* adaptation process. More precisely, the use of every *morph agent* at each organisation cycle is the main concern. A solution could be to give priority to critical *morph agents* while non-critical ones could not adapt. In this case, a critical *morph agent* would be unsure about its transformation function.

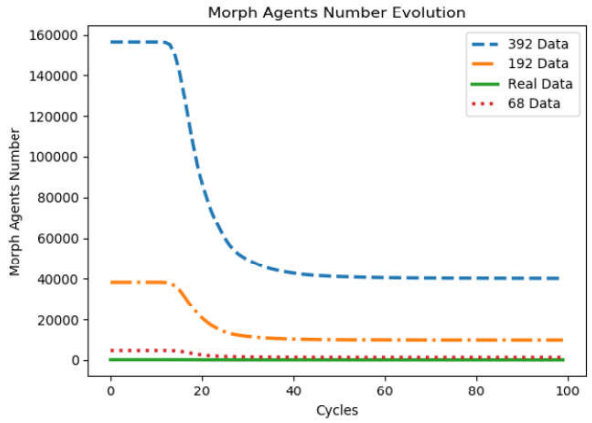
$$complexity = \begin{cases} \frac{nbData^2 \times 3}{nbParallelAgents} & \text{(worst-case)} \\ \frac{nbData \times 3}{nbParallelAgents} & \text{(best-case)} \end{cases} \tag{5.2}$$

The peaks that can be seen in figure 5.8 illustrate the worst case that may happen in a given environment. The variations after the 30^{th} cycle are massively related to the computer's performance (because the number of *morph agents* is stable and the history size is limited). As explained in the chapter 4, to avoid useless agents using resources, a *morph agent* has the possibility to destroy itself.

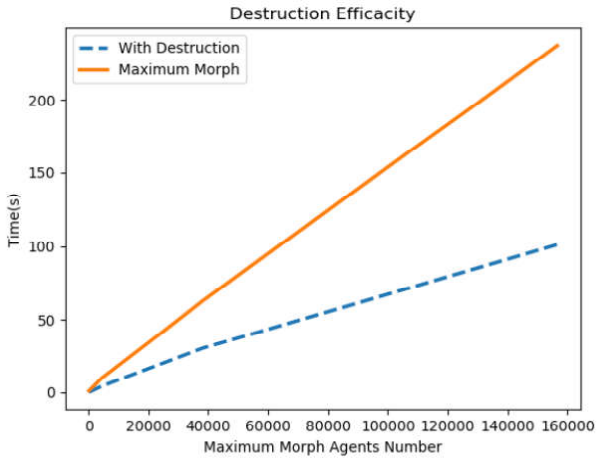
Figure 5.9a shows the evolution of the number of *morph agents* during the execution. At first, the number of *morph agents* is equal to the square of the *data agents* number. Then after a few cycles, around 17, the number reduces quickly to then stabilise. This value depends on the number of related data between them. More data represent the same information, meaning it exists a transformation function between them, more *morph agents* are kept alive. This figure does not show variations after the stability stage is reached. This reinforces the statement that the computer is responsible for the variations.

Figure 5.9b shows the correlation between computation time and the number of *morph agents*. By reducing the number of *morph agents*, the overall computation time is drastically lowered. The increase sees its slope reduced, thus enabling more time between two cycles. As no agent is parallelised, more reduction is possible depending on the available resources and hardware.

To conclude on these three experiments, the LUDA module shows an appropriate behaviour towards high dimensional problems. Some improvements may be done by adding another cooperative strategy to lower



(a) Evolution of the adaptation of the coefficients of *morph agents* number



(b) Difference between keeping *morph agents* and enabling them to destroy themselves

Figure 5.9: Number of *morph agents* evolution.

unnecessary computation. Thanks to the distribution of tasks among the three types of agents, it is possible for a *group agent* to send a value if necessary while some *morph agents* and *data agents* are in the organisation phase. Giving the priority to the most critical agents seems to be a good improvement to do in the future.

As *morph agents* are the main resources and time consumers in the system, another solution to reduce time consumption is to reduce the time taken by their behaviour. The most time-consuming part of their algorithm is the examples processing (the only loop that increases). The next section discusses how the number of historic influences the *morph agents'* behaviour.

Study on the Observations Number

A strong emphasis is put on the examples and more precisely on the number of observations needed to adapt the transformation function of *morph agents*. An observation is stored as an example in a *morph agent'* history. An observation includes both *morph agents'* data (creator and objective) value that are available at the same time. In large environments with many examples and many *morph agents*, the number of observations grows exponentially. To avoid the exponentially growth, a *morph agent* is able to self-destruct when it understands that it is useless. But to be aware of the necessity to destroy itself, it needs to adapt its transformation function.

Figure 5.10 illustrates the impact of the number of observations kept in the history of a *morph agent*. It can be seen that an increase in the number of observations reduces the peaks intensity while keeping the mean error low. However, no significant improvement is observed after 15 observations. This figure suggests to limit the number of observations to 15, thus limiting the reserved memory.

Figure 5.11 shows the number of cycles required to see a reduction of the number of *morph agents*. As before, this experiment has been realised by giving every data at every cycle from the dataset. This figure shows that 5 observations are not sufficient to the system to converge. But, with a higher number of observations, several *morph agents* choose to self-destroy. A bound seems to be reached with 20 observations with nearly no difference between 20 and 25 observations.

These two previous figures, 5.10 and 5.11 illustrate the impact of the number of observations required for the good functioning of *morph agents*. This number stays stable with different datasets excepted for small ones. These small datasets require less observation but they are the easiest ones. Because a *morph agent*

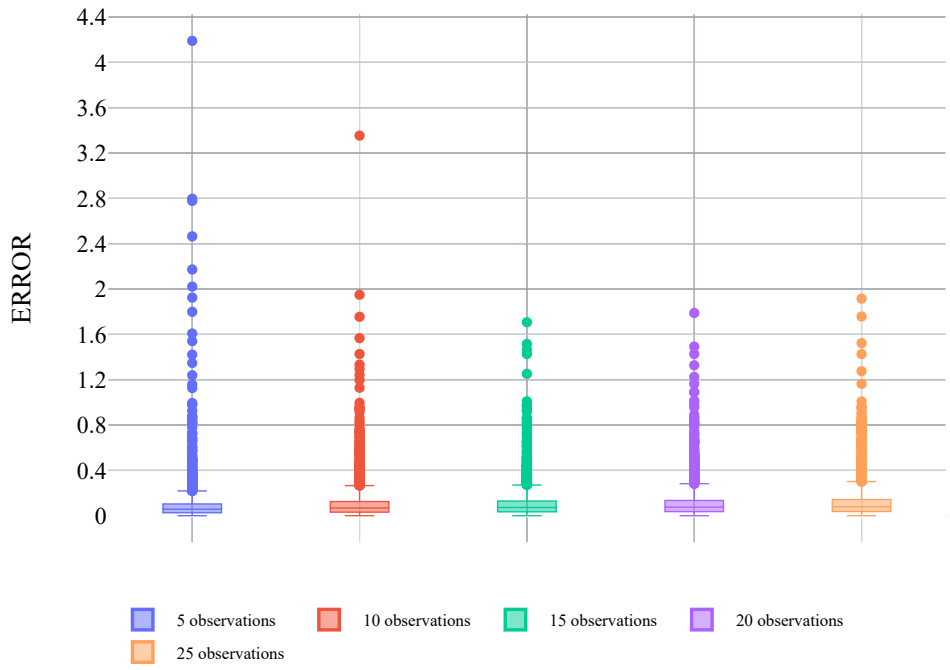


Figure 5.10: Error between e_d and p_d according to the number of observations

does not need to keep many observations in history to operate it has a low memory requirement. This means that depending on the problem constraint, the module can be used with 10 observations by *morph agents* with maximum efficiency with 20 observations. This is a parameter that can be calibrated from the addressed problem's requirements.

The second organisation step that may be an issue for the computation time is the *group agent* organisation, specifically when a *data agent* needs to be excluded or shared. The negotiation time of every *group agents* negotiate can be high. However, because the number of *group agents* is low, they can easily be parallelised. Another way is to reduce the computation time of *group agents* is to limit their view of other *group agents*. This allows addressing bigger problems. In this eventuality, *group agents* would only perceive a few number of *group agents*. So, it would be more difficult for a *data agent* to get to the correct *group agent*. The solution for *group agents* would be to incorporate bad *data agents* only, to give them to another *group agent* that was previously not in its neighbourhood. For example, suppose three *group agents* (ga_0, ga_1 and ga_2). The *data agent 1* (da_1) is in ga_0 and its ideal *group agent* would be ga_2 (da_1 does not know it). ga_0 can perceive ga_1, and ga_1 can perceive ga_0 and ga_2. To get to the ga_2, first ga_0 needs to give da_1 to ga_1 and then, ga_1 has to give da_1 to ga_2. This point is not in the scope of this thesis but it needs further investigations.

5.2.5 Synthesis on LUDA's Experiments

The LUDA module has been designed to enable an agent to transform communicated information in another referential to make it usable. After describing it in the previous chapter, this section has focused on results obtained thanks to multiple experiments with the LUDA module that are analysed using several metrics and tools. With regards to the challenges of **scalability** and the efficiency to transform an information, the resulting computation time and adaptability are efficient. First, the section has presented how experiments are realised, and has exposed the notion of abstract black box generator. This AMAS-based generator enables the LUDA module to be plugged into a black box. After explaining the abstract black box generator architecture,

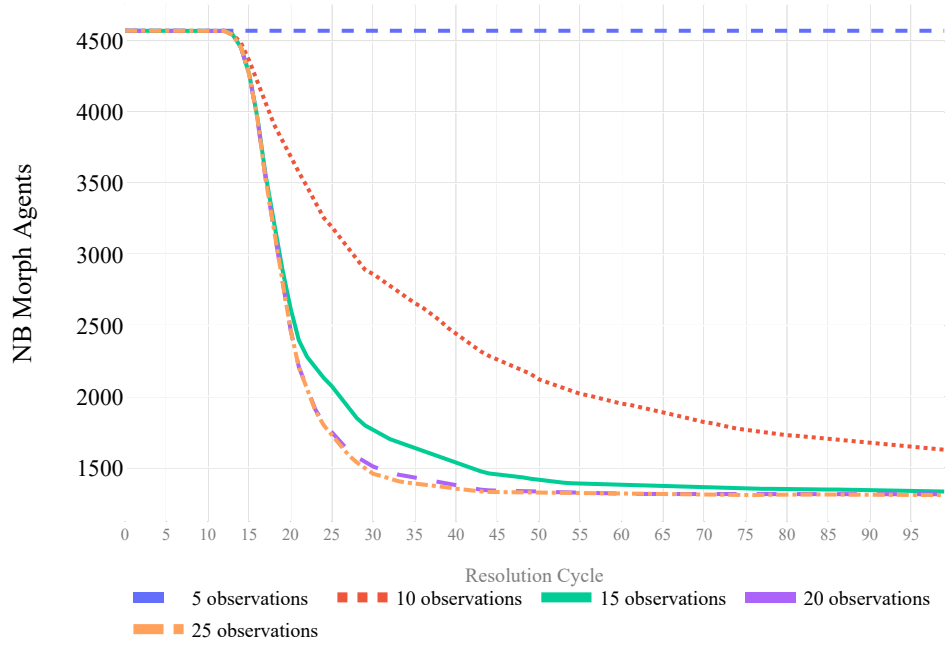


Figure 5.11: Evolution of the *morph agents* number according to the number of observations

the experiments are detailed. Experiments are divided into 3 types meant to analyse the module's behaviour in specific ways.

The first step presents the module's cooperative process to provide an estimation for data to the black box. Plugged into the black box, the LUDA module receives the abstract black box's perceptions, and all agents in the LUDA module self-organise. To evaluate this, the resulting groups are observed. Results show an adequate organisation of the module. From the correct organisation, the module can provide a near-optimal answer to missing data. Experiments in a noisy environment present that the final organisation is not always perfect but the module is still able to provide a value to the abstract black box inputs while in an under-performing way.

From the previous organisation evaluation, the module is able to provide an estimation for missing data. This second step compares with state-of-the-art data estimation methods. Using several datasets with different characteristics (as the number of data involved), different generation methods, and different noise values, results of each method are analysed. The LUDA module takes advantage of the massive amount of data to obtain a better estimation for data.

Last, the LUDA module's characteristics are explored to study the **scalability** challenge matching. The use of AMAS makes the module suitable for the **openness** characteristic. Indeed, adding a new information in the environment is immediately handled with the creation of a new agent and does not disrupt others. The **scalability** is evaluated by looking at the computation time necessary to realise one organisation cycle. The use of a self-destruction strategy enables to reduce drastically the computation time. Moreover, this reduces the impact of the increase in scale.

Integrating this module into a CAV's ADP is easy as it is not specialised in a specific data type. Depending on the requirements and resources available, the LUDA module can be calibrated according to those. Thanks to it, it is possible to focus on the communication optimisation problem.

5.3 Communication Optimisation Experiments

This section presents the experiments done to validate the CODCOP method. As in the LUDA module experiments, two phases are used. The first one enables to compare CODCOP's results with DCOP state-of-the-art methods to validate its efficiency. This allows to position it among these state-of-the-art methods thanks to the DCOP taxonomy. The second phase consists in evaluating the proposed method to validate the required characteristics of **dynamicity** and **scalability**.

In this section the experimental conditions and scenarios are first explained. Then the state-of-the-art DCOP methods are used on "simple" scenarios to have their results compared to the CODCOP one. After that, much-complicated scenarios are used exclusively on the CODCOP module to check its adequacy with regard to dynamicity and scalability. Finally, a synthesis is provided.

5.3.1 Experimental Process and Metrics

In this section, the experimental conditions used to evaluate the CODCOP module are presented. The use of Python 3 as a programming language is justified, the experimental process used is described according to the *pulp* solver and the *pydcop* library, and finally, metrics used to evaluate the module are presented.

The PyDCOP Library

Many available generic algorithms are able, given a proper DCOP formalisation of a problem, to solve the problem. When evaluating a Dyn-DCOP algorithm, time is an important metric. The algorithm's quality is not the only factor influencing the time. The programming language and hardware also influence the results. To obtain an evaluation without bias it is important to use the same environmental conditions for every algorithm to compare. Recently, a library written in python 3, called pyDCOP, provides an implementation of many state-of-the-art DCOP algorithms, but no ADCOP ones.

The pyDCOP library proposes Maximum Gain Message and its variant Maximum Gain Message 2. Unfortunately, the extensions MCS-MGM and GCA-MGM are not proposed. But this library has motivated the design of the CODCOP algorithm using the Python 3 language. PyDCOP uses a special file type, called *yaml*, to transcript a DCOP problem. Giving this file to a PyDCOP's algorithm, this one begins to solve the corresponding problem. However, many characteristics must be specified:

- The global objective function, either minimising or maximising it.
- The name of all agents.
- The domain of each variable.
- All the soft constraints, including a cost, the possibility to write a python function. This allows writing conditional functions and mathematical functions.
- The cost of communications between agents.
- With an additional file, it is possible to specify events: adding/removing an agent or modifying variables and constraints.
- With an additional file, it is possible to specify the variables that are possessed by each agent. This is useful if the assumption of only one variable by an agent is made.

Experimental Process

The CODCOP module may be evaluated in two ways: it can be run with or without requiring a special file format as input. If the *yaml* file is not given a simple text file is required as input. The AMAS extracts the problem's characteristics from the text file. If a characteristic is missing, a default value is used. While solving it, the CODCOP module creates an equivalent *yaml* file that can be used by the other state-of-the-art algorithms. If the *yaml* file is given as input, CODCOP extracts the information from the file. Each experiment is described by the following characteristics:

- 1) the initial number of agents nb_agents. With each different created agent, an equivalent number of variations for each data is created.
- 2) the number of different information in the environment nb_inf. This corresponds to the maximum number of different information a problem can have. Each agent creates its variation from it. Then it associates with it the personal utility. The utility is calculated using a Gaussian Distribution around the default value.
- 3) the default max processing capacity $max\_process$ of agents. This is directly related to the agent's hard constraint.
- 4) the possible variation of the max processing capacity $max\_process$. This is used to differentiate agents to not have to solve a problem where all agents are homogeneous. This characteristic gives a maximum offset percentage from the default max processing capacity. The $max\_process$ of an agent is then calculated using a Gaussian distribution on it.
- 5) the probability to have an agent entering the system. At each end of a cycle, a random number is generated. If the value is lower than the probability then a new agent is created. The new agent is placed in a way that it has at least one agent in its neighbourhood.
- 6) the probability to have an agent leaving the system. Similarly to 5), a random number is generated. If the value is lower than the probability then an agent is removed, and the system ensures that the graph is still connected.
- 7) the probability to have an agent modification. However, with a random chance, an agent may be moved in another location or the value of its variables modified. The resulting graph has to be connected.
- 8) the communication range, that directly impacts the neighbourhood size of each agent. When the communication range is global, this means that each agent is a connected agent, thus the graph is complete.

All algorithms are run over 300 cycles. A cycle begins with the synchronisation of all agents and ends when all agents have acted. The number 300 (for cycles) has been chosen to include both a sufficient resolution time, the possibility to add multiple dynamic changes, and to compare how each algorithm modifies its local solution.

Table 5.3 presents the characteristics of all scenarios used to experiment with the module. In these scenarios, the dynamic characteristics are not taken into account because perturbations are added independently of the chosen scenario. Scenarios 0 to 6 are used by both state-of-the-art algorithms and CODCOP. Scenarios 7 to 11 are only used by the AMAS as standards algorithms require too much time to find a solution.

	Nb_agents	Nb_var	Ccom	Ccom_variation	Com_range
Scenario0	3	3	3	0%	Global
Scenario1	3	30	20	0%	Global
Scenario2	3	50	25	10%	Global
Scenario3	6	50	40	0%	1
Scenario4	6	60	40	25%	Global
Scenario5	3	100	50	25%	Global
Scenario6	9	50	25	10%	Global
AMAS Only					
Scenario7	9	100	50	25%	1
Scenario8	9	1000	500	25%	Global
Scenario9	30	1000	500	25%	Global
Scenario10	50	1000	500	25%	1
Scenario11	50	1000	500	25%	Global

Table 5.3: Experimental scenarios used to evaluate the CODCOP module.

The process used, for the first type of experiment, is described in figure 5.12. From scenario characteristics, the AMAS creates 3 different files: a file called ”results AMAS” (value of objective function and variables assignment at every cycle), a file called ”experiment.yaml”, with information needed by the solver and DCOP algorithms, and a file called ”events” with dynamic modifications. The *Pulp* solver is used to find the optimal solution for the problem. *Pulp* is a linear solver (https://coin-or.github.io/pulp/) able to find the optimal solution in a linear time. From created documents the pulp solver is run as well as DCOP algorithms for each DCOP to solve. However, when the number of variables and agents increases, some results may be difficult to obtain. This is why only scenarios 0 to 6 are used for comparison. In the other scenarios (7 to 11), the complexity increases so that it requires days to get a result. Finally, all the results are aggregated into a single file and compared together. MGM and MGM2 algorithms used for comparison are from the library pyDcop [130]. MCS-MGM and GCA-MGM have also been implemented following [51], but they can’t be used on the addressed problem. PSyncBB has not been used because the computation time required is too high.

Metrics

Comparison is made using the following common metrics:

- The **total cost** of each algorithm’s results. The first step is to evaluate the algorithm that maximise the best global objective function. Thus, each scenario is run 100 times and results are aggregated to avoid a lucky shot.
- The **time to achieve one cycle** and provide a solution. This is related to the dynamic challenge. The algorithm has to be able to provide a solution if necessary, even a non-optimal one.
- The **time (number of cycles)** until the solution’s convergence. An algorithm has converged when all agents stop modifying their variables’ value.
- The **number of messages used to solve the problem**. The objective is to optimise the communication, because using too many messages to solve the problem is counter-productive.

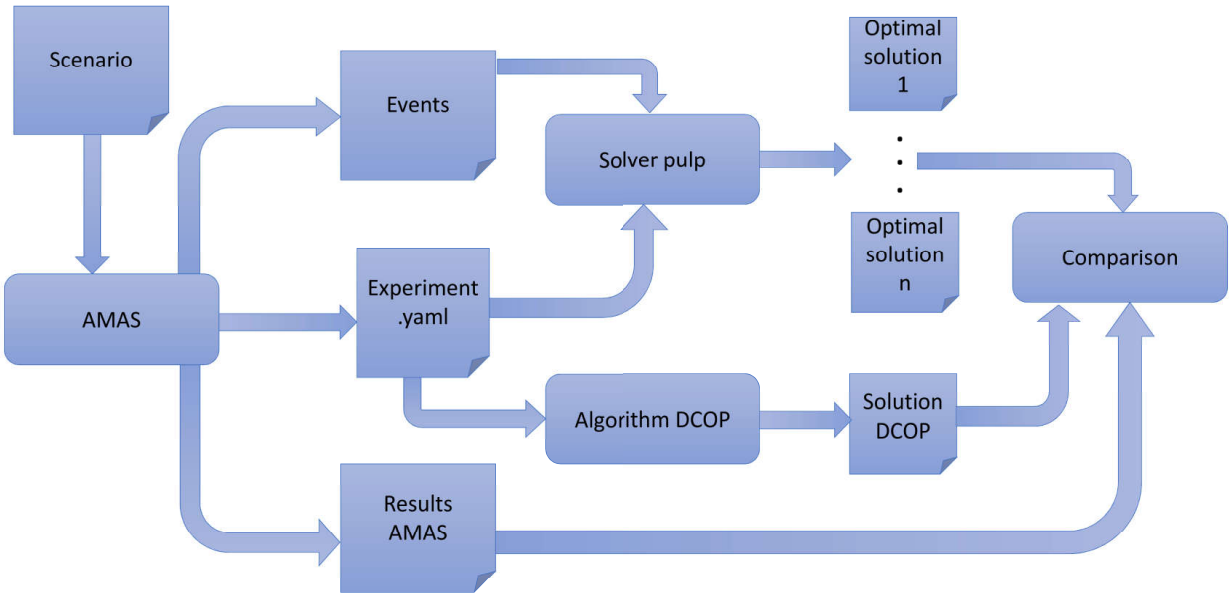


Figure 5.12: Evaluation process

After the metrics are used to compare the result, the AMAS behaviour needs to be evaluated with regard to dynamic and scalability challenges. More metrics and experiments are used to evaluate the AMAS.

- The cycle computation time evolution between different experimental conditions, particularly the number of variables. If this one increases too quickly, scalability will be limited.
- The time to converge after a modification is taken into account.
- The quality of the solution after a modification is taken into account.

This exploration is done with the scenarios presented in table 5.3 but also with self-generated scenario as stated in 5.3.1. Thanks to this, it is possible to have an infinite number of different and non-controlled scenarios available.

5.3.2 CODCOP and DCOP Algorithms Comparison

To give a comparison between the CODCOP module and state-of-the-art methods, these latter are run on the scenarios. This aims at giving an idea of their limit for dynamicity and scalability challenges. In table 5.4 algorithms' results are presented after 300 cycles. Note that the scenarios names are reduced to facilitate the reading (for example, "scenario0" become "sc0"). In this figure, each algorithm has been run on each scenario 100 times. All results have been aggregated and the average maximum cost is displayed. As selected state-of-the-art ADCOP algorithms are not designed specifically for dynamic problems, no dynamic changes are integrated into the experiments.

The table 5.4 shows the resulting costs from the variables' assignment decided by the algorithms. With these examples, an optimal solution is found by the pulp solver. As expected, algorithms MGM and MGM2 are not able to find a near-optimal solution, the resulted cost is far from the solver's one. They are not suitable for asymmetric DCOP. In addition to the cost, the amount of messages is indicated in the table. This

	MGM				
	cost	msg_count	Time (s)	cycle time (s)	Pulp
Sc0	8.5	32382	3.28	0.010933333	62.48
Sc1	6.76	2955545	257.92	0.859733333	539.28
Sc2	4.98	8753909	693	2.31	774.21
Sc3	10.91	43451611	4157	13.85666667	1331.38
Sc4	4.02	34308202	2836.39	9.454633333	1453.94
Sc5	4.94	34779406	3065	10.21666667	1515.91
Sc6	18.13	38249367	4714	15.71333333	1414.90
	MGM2				
Sc0	29.47	49672	1.97	0.006566667	62.48
Sc1	20.47	4444708	312	1.04	539.28
Sc2	23.95	13146152	819.66	2.7322	774.21
Sc3	21.31	65199520	3745	12.48333333	1331.38
Sc4	20.02	51471636	3052	10.17333333	1453.94
Sc5	24.8	52177868	3586	11.95333333	1515.91
Sc6	29.44	57401380	3579	11.93	1414.90

Table 5.4: Evaluation of MGM and MGM2 DCOP algorithms

amount is bigger than each agent's computational capacity. Using as many messages as these algorithms is problematic because the hard constraint is directly impacted by the number of communication. Considering the computation time, the cycle time increases fast with the problem dimension. This means that they are not suitable for high dimensional problems. So, scalability is not ensured with these state-of-the-art algorithms.

MCS- and GCA-MGM results are not presented because the problem's characteristic that prevents the constraint exchange strategy used by these algorithms:

- When an agent A chooses a value for a variable that results in further degradation of the solution for another agent B, the agent B sends its side of the constraint (the variable impact). So the agent A receives the information about how its variables impact someone else. In the CODCOP context, it is not suitable for both constraints types. The hard constraint is impacted according to every variables excepted for the one owned by the agent. It is impossible for an agent to modify its variables' value without knowing everything about its neighbours. Second, the maximum functions is chosen to compute the *soft constraints* cost, it is impossible for an agent without all the knowledge to understand if it must put its variables to 1 or 0. As a consequence, to have MCS- and GCA-MGM to be efficient, each agent should know everything about others.

- This problem has a particularity: agents' own several variables. So, the common assumption of only one variable by agent is not checked. This has a negative consequence on state-of-the-art algorithms. Indeed, MGM extensions only take into account one variable. The local reduction changes only one variable at a time. According to the same strategy, the number of local reductions necessary increases with the increase of the number of variables without increasing the number of agents, thus increasing the computation time.

Table 5.5 contains the same scenarios as table 5.4 for the AMAS proposition with additional scenarios (7 to 11). We can notice that the CODCOP module is able to obtain a solution with a total cost better than other algorithms. The time taken by the algorithm to converge to a solution is low. It is able to give a solution

	CODCOP				
	Cost	Message volume	Time (s)	cycle time (s)	Pulp Solver Cost
Scenario 0	40	54	0.268044	negligible	62.48
Scenario 1	297	67	0.276770	negligible	539.28
Scenario 2	498	77	0.291848	negligible	774.21
Scenario 3	966	324	0.866442	negligible	1331.38
Scenario 4	911	113	0.619276	negligible	1453.94
Scenario 5	978	62	1.094064	negligible	1515.91
Scenario 6	1236	524	0.757063	negligible	1414.90
Scenario 7	3275	1152	1.308537	0.00436	Unavailable
Scenario 8	33473	153	22.814220	0.0760	Unavailable
Scenario 9	120128	424	74.151558	0.24717	Unavailable
Scenario 10	125154	1202	105.566179	0.3519	Unavailable
Scenario 11	211825	787	118.387235	0.3946	Unavailable

Table 5.5: Evaluation of the AMAS on scenarios

in an acceptable time for scenarios where the pulp solver resolution is too long. More investigation on each characteristic is done in the next sections. A main limit about the contribution is the difficulty to compare with state-of-the-art ADCOP algorithms. The problem is too specific and the constraints' nature prevents comparing with the state-of-the-art algorithm. MGM and MGM2 can be used on the problem but are not efficient due to the asymmetric characteristic, which validates the DCOP formalism.

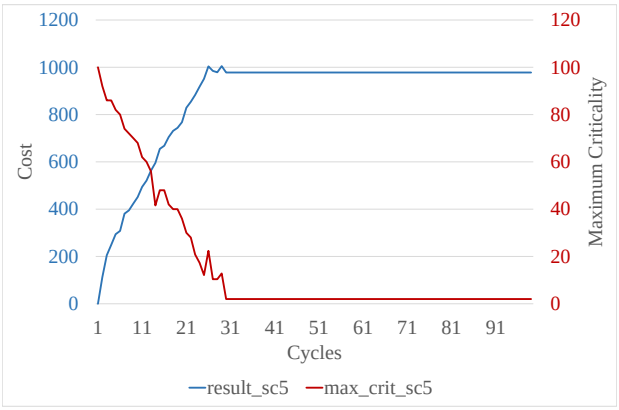
5.3.3 CODCOP Exploration

Because a 100% accurate comparison cannot be done between state-of-the-art algorithms and the proposed solution, this section and the next one aim at providing the reader with the most insight about the system's characteristics. This section presents experiments that help to assess the **scalability** challenge. First the relation between the maximum criticality in the system and the system convergence is given (Part 1). Thanks to this, it is possible to explore characteristics of high dimensional problem. Then the increase of the cycle's time is studied (part 2), followed by the number of negotiation messages evolution. Finally, the total cost evolution is provided.

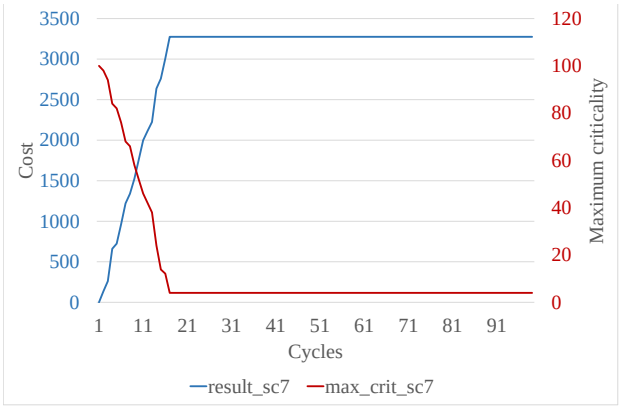
Part 1 - Relation between Criticality and Objective

Before going deeper in the exploration, the first step is to understand how agents' criticality affects precisely the overall result. The motivation is to facilitate the evaluation by only observing the agents' criticality. Figure 5.13 shows the relation between maximum criticality and global result (only 100 cycles can be seen for visualisation purposes). We see a strong correlation between the diminution of the maximum criticality and the increase of the global result. Until the criticality convergence is not yet achieved, the adaptation can lead to the violation of a constraint. However, once the convergence is achieved, no constraint is violated. This is ensured by the fact that the maximum negative criticality for an agent is achieved when its $ccom$ is violated. We can also see that if a convergence is not achieved, then the global result is not stable (figure 5.13c).

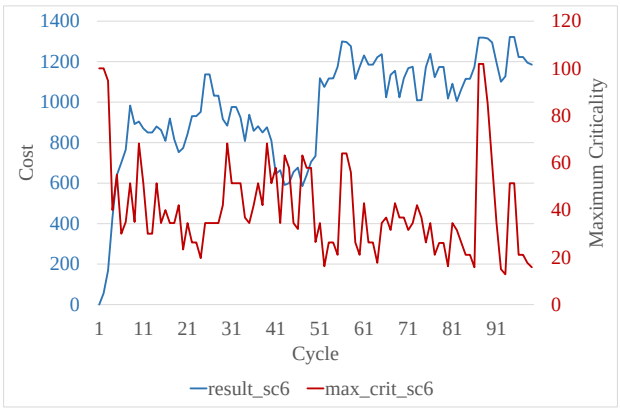
This experiment investigates how the CODCOP module behaves with different communication ranges. A



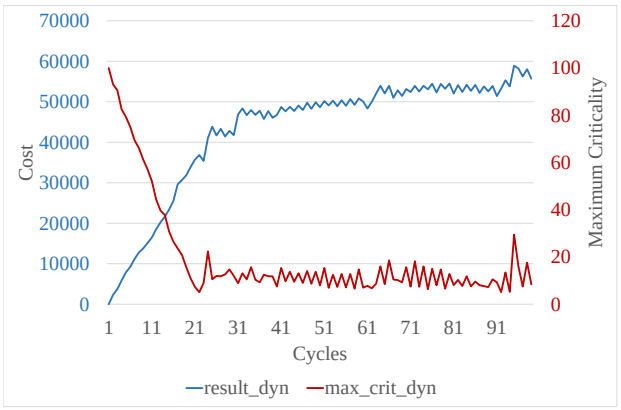
(a) Evolution of global result on scenario5



(b) Evolution of global result on scenario7



(c) Evolution of global result on scenario6



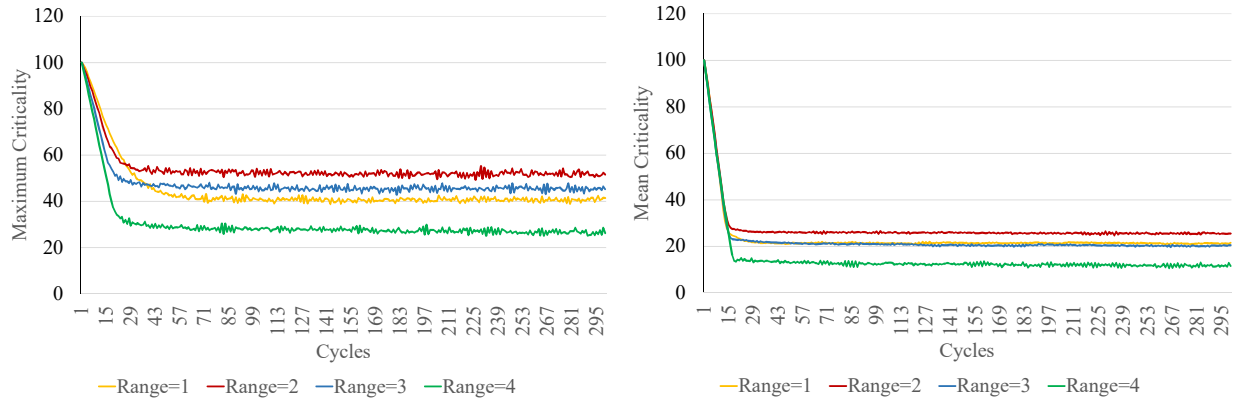
(d) Evolution of global result on a dynamic scenario

Figure 5.13: Evolution of global result according to criticality

wider communication range means a more connected graph. This has a direct influence on soft constraints' variables and also the possible total amount of shared massages. Looking at the criticality provides a deeper understanding of the overall influence of the communication range. This is quite effective for high dimensional problems where no truth can be obtained. Figure 5.14 shows both impact on the mean criticality and the maximum criticality on all agents in the system. Because the problem constraint is related to communication between agents, each exchange between agents increases the burden on the agent computation. By taking into account the limitation of communication range, agents do not know how their action impact all their neighbours. Looking at results presented in figure 5.14, the maximum criticality with limited communication range is higher than with a global overview. This is due to the fact that agents with small neighbourhood get less variables than agents with a dense neighbourhood. An improvement would be to take into account the density of the neighbourhood to decide the value of a variable. From this, a criticality about the neighbourhood density can be designed to encourage specific agents to modify their variables. However, doing so would create a new resolution message transiting through the network. Moreover, neighbourhoods can be modified really quickly and such a strategy needs to take into account this change.

Part 2 - Cycle Time Exploration

Computation time is the most important criterion for critical systems as well as finding an adequate solution. In critical systems, a time is considered as reasonable if it is less than a critical threshold. A real-world application



(a) Evolution of maximum criticality with different communication range

(b) Evolution of mean criticality with different communication range

Figure 5.14: Evolution of global result according to criticality

like road traffic can be subject to any time threshold. It is difficult to assess whether an algorithm is effective for any situation, but, it is possible to analyse the response time of the algorithm, even without knowledge about the number of information and computation capacities of a vehicle. The experiment presented hereafter considers the evolution of the computation time of an agent's cycle according to the number of information and the number of agents. In CODCOP, a cycle begins with a synchronisation phase of all agents and ends once all agents have acted.

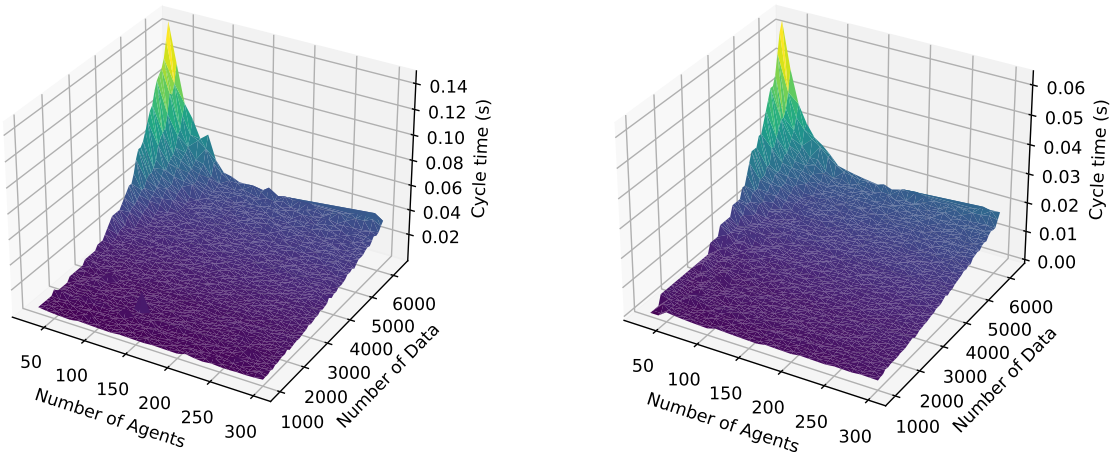
Both figures 5.15a and 5.15b show how on agent's cycle time evolves according to the number of neighbouring agents and the information volume available in the environment. These two figures tends to have the same behaviour. It can be noticed that the **maximum computation time is reached** when the **number of agents is the lowest and the number of information the highest**. More information to process leads to more time to achieve a local decision. Concretely, an agent needs more time to decide what value to give for each one of its variables. In the worst-case scenario, with 30 agents and 6300 information, the maximum cycle time of an agent reaches only 0.14 seconds. For most scenarios, a cycle time only reaches 0.03 seconds, meaning that it can theoretically compute 33 cycles within one second. This computation time is even lower when looking at the mean processing time.

In both figures, it can be observed that the **number of agents has a direct consequence on the processing time**. With a reduced number of agents, each of them has statistically more chances to increase the number of information to share. Consequently, it has to take more time to decide what value to give to each one of its variables. However, these experiments do not take into account the communication time for every agent to synchronise together. In a real application, this communication time has to be considered according to the network characteristics, reducing the number of cycle possible to do.

Part 3 - Problem Total Cost Evolution

The total cost of the CODCOP module when it has converged is now evaluated. For the explored problems (many scenarios with different number of agents and number of data) it is impossible to know the ideal assignment and thus the optimal solution. This is due to the solver taking too much time to find the solution. However, the CODCOP module behaviour on larger problems enables to know if the system is not stuck in a local maximum.

Figure 5.16 shows the evolution of the total cost according to the number of agents and the number of data



(a) Evolution of the maximum cycle time according to the number of agents and the number of data

(b) Evolution of the mean cycle time according to the number of agents and the number of data

Figure 5.15: Evolution of the cycle time according to the number of agents and the number of data.

in the environment. The increase of both dimensions (number of agents and number of variables), leads to an increase of the total cost. The computation cost for each agent increases with the number of data. According to previous figures exploring the computation time and the total cost evolution with bigger dimensions, the characteristic of **scalability** is checked. The next section explores the CODCOP behaviour on the **dynamic** and **openness** characteristics.

5.3.4 Dynamic and Openness Compliance

Experiments presented hereafter show how the CODCOP module reacts to environment modifications. Each possible modification is evaluated individually. A global communication range is used for these experiments. By doing so this, a perturbation will have the strongest impact on the overall system.

This section studies first the characteristic of **openness** by looking at the impact of adding and removing an agent from the system during its execution. Then the impact of dynamic changes is observed by bringing disturbance in the environment. Finally, a conclusion is made on the **dynamic** and **openness** characteristics.

Arrival of new agents

Three different types of additions are studied:

- a) The addition of a better agent. This means that the new agent has a maximum processing capacity ($max\_process$) bigger than the ones in the environment. This experiment allows to observe that even with the introduction of such an agent, no hard constraint is violated.
- b) The addition of a new agent with the same maximum processing capacity ($max\_process$) as the others. The objective is to observe a modification in the communication distribution.
- c) The introduction of an agent with a low maximum processing capacity. This should cause disturbances and leads many agents to modify their decision.

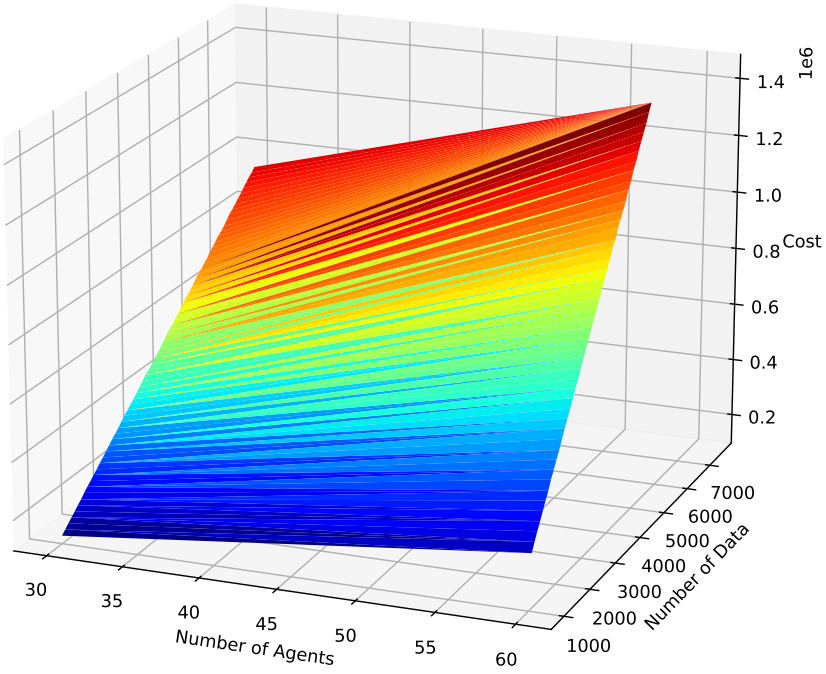
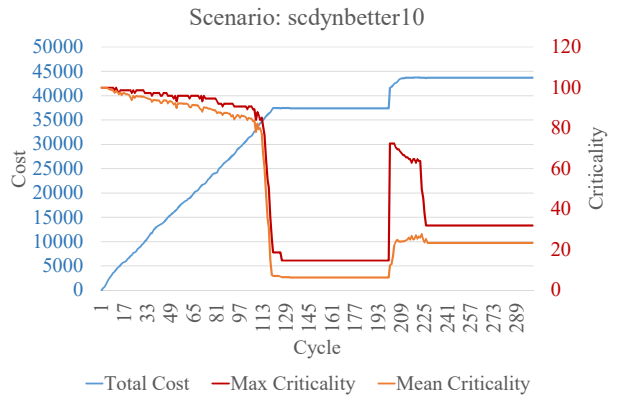


Figure 5.16: Evolution of the total cost according to the number of agents and the number of data

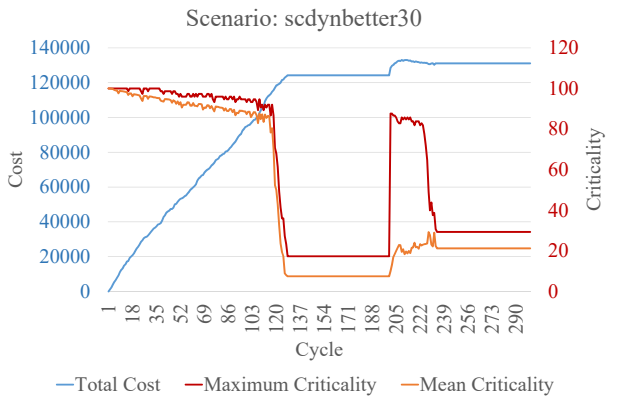
Each experiment is named using scenarios according to the format: ”scdyn” + <agent type> + <number of initial agents>. Agent type includes three values: ”ident”, ”better” and ”worse” depending on the arriving agent’s characteristics. Two initial numbers of agents are used for the experiments: 10 and 30. The aim is to increase the number of agents and to observe the consequences of this increase. For example, an experiment with 10 initial agents where an identical agent comes is named ”scdynident10”. Excepted the arriving agent, each agent has a $max\_process = 500$ during the experiments.

Figures 5.17a and 5.17b illustrate the impact of the arrival of a better agent with a bigger maximum processing capacity (700 instead of 500). The arrival of this new agent in the environment creates a perturbation that increases the criticalities that were stable for more than 50 cycles. After several cycles, both criticalities converge again. However, the value of the criticality convergence is higher than before the perturbation, because the criticality computation depends on the maximum processing capacity. As the new agent’s maximum processing capacity is higher, its criticality is higher too (it receives fewer messages in proportion to its maximum processing capacity, thus increasing its criticality linked to its desire to receive more messages).

A second observation can be made in figure 5.17a. The cost achieved is higher after the introduction of the new agent. It does not reach the maximum immediately, but while the criticality is not stable, the total cost is continuously increasing. During this unstable period, the agents that exchange an information are changing until an agreement is found. In figure 5.17b, the maximum total cost is not the cost obtained after the convergence. There are two major reasons behind this sub-optimal result. The first one is because of the local preferences of agents. Because an agent never shares its local preferences, it may happen that an agent withdraws its communication whereas it was the best. The second reason is related to the calibration metric. If the number of messages is equal (or almost equal) to the maximum processing capacity of an agent, then the hard constraint (the number of received messages must not exceed an agent’s $max\_process$) can be violated with further adaptations. When violated, the system quickly adapts to reach a state with no violation. But a hard constraint should not be violated during the optimisation process, except due to the environment

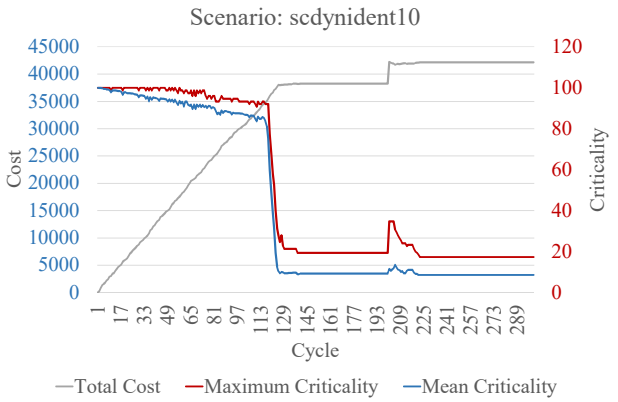


(a) CODCOP behaviour with the addition of a *better* agent at the 200^{th} cycle with 10 initial agents.

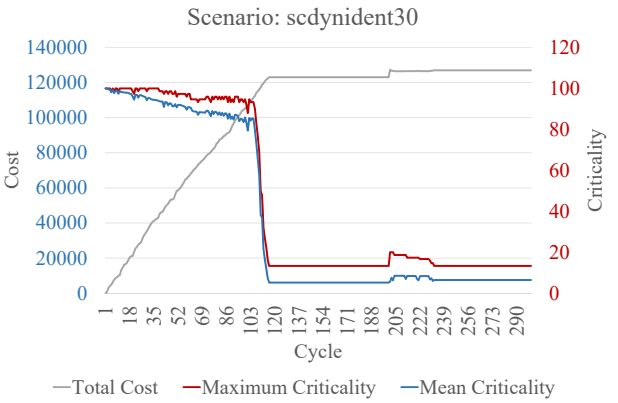


(b) CODCOP behaviour with the addition of a *better* agent at the 200^{th} cycle with 30 initial agents.

Figure 5.17: System evolution with the arrival of a *better* agent



(a) CODCOP behaviour with the addition of an *identical* agent at the 200^{th} cycle with 10 initial agents.



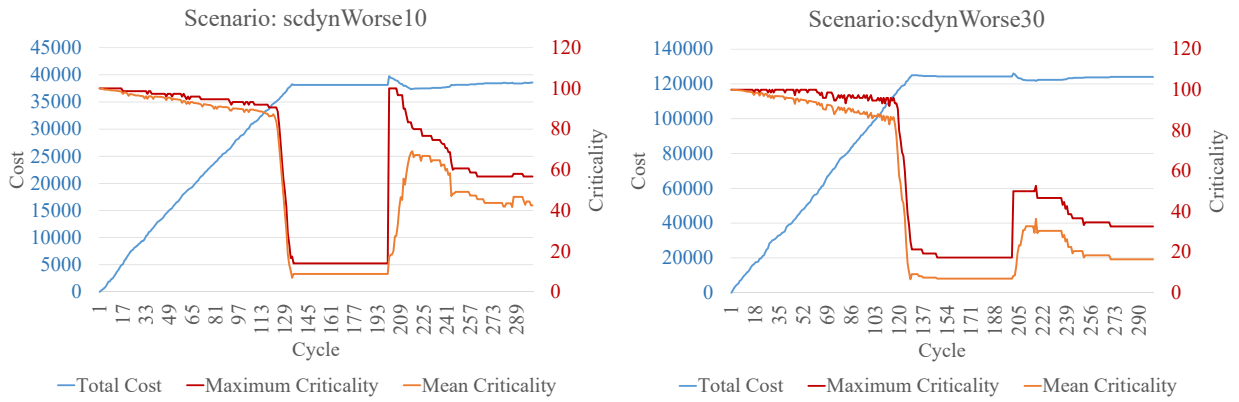
(b) CODCOP behaviour with the addition of an *identical* agent at the 200^{th} cycle with 30 initial agents.

Figure 5.18: System evolution with the arrival of an *identical* agent

modifications. Such a modification may put the system into a non-functional state independently of the agents' behaviour. Because modifications cannot be anticipated (or at least only partially), a user can calibrate the system to lower the danger with a negative impact on the total cost reached by modifying the thresholds in the criticality computation.

Figures 5.18a and 5.18b illustrate the impact of the arrival of an agent with the same maximum processing capacity as others (the Gaussian distribution is kept). A perturbation is seen when the new agent is introduced illustrating the modification on the social environment of a CAV. While the introduction of a better agent increases the overall criticality in the system, the introduction of an identical agent,hardly changes the criticality.

Figures 5.19a and 5.19b illustrate the impact of the arrival of an agent with a worse maximum processing capacity (300 instead of 500). When the agent is added, its hard constraint is immediately violated. This is illustrated with the peak criticality score reaching 100 at the 200^{th} cycle in figure 5.19a. However, in a few cycles, the criticality decreases meaning that the hard constraint is no longer violated. The total cost increases with the introduction of the new agent. When a new agent is added in the system, the maximum total cost



(a) CODCOP behaviour with the addition of a *worse* agent at the 200^{th} cycle with 10 initial agents.

(b) CODCOP behaviour with the addition of a *worse* agent at the 200^{th} cycle with 30 initial agents.

Figure 5.19: System evolution with the arrival of a *worse* agent

possible increases. The total cost shown in figure 5.19a does not take into account the hard constraint, which should reduce the cost by ∞. Because the new agent maximum processing capacity is lower, neighbours agent reduce the number of variables whose value is 1 and a decrease of the total cost is observed a few cycle after. However, with more cycles the total cost increases once again.

The introduction of a new agent has an impact on the system state. Depending on the agent's characteristics, this latter has a criticality value that is often worse than the ones of other agents. In consequence, the system enters a state of re-organisation until it converges once again. A hard constraint can be violated if a new agent enters the system and if the number of the message exceeds its maximum processing capacity.

The next experiments study the system behaviour when an agent leaves the system.

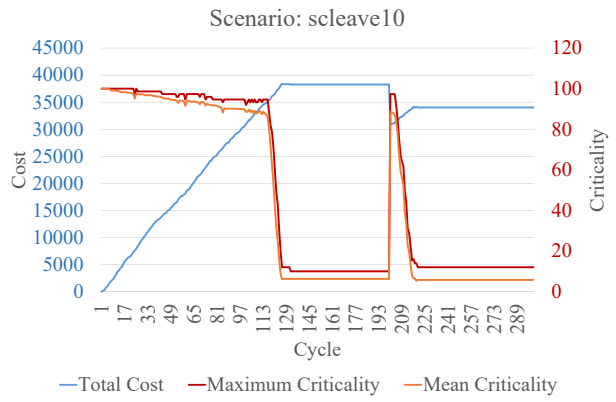
Leaving Agents

An agent that leaves the system should soften the hard constraints (because the leaving agent's messages are no longer exchanged) but many variables are removed because of its departure. In consequence, the overall criticality is likely to increase, except if the leaving agent was the most critical. In the following experiment, an agent is removed at the 200^{th} cycle.

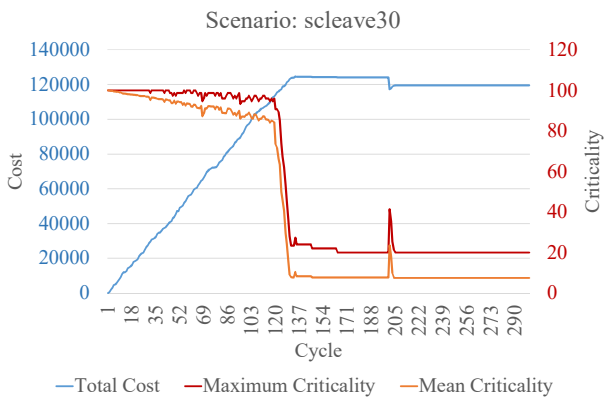
Both figures 5.20a and 5.20b illustrate the system's behaviour confronted to the departure of an agent. As expected a peak is seen when the agent is leaving because it does no longer communicate. Thus, the whole system is impacted and the first sub-criticality increases (representing the distance of the ideal number of received messages and its maximum processing capacity). In the following cycle, other agents share more messages in the system until reaching a new stable state. With the agent departure, the overall criticality is lower than before. The impact of a leaving agent is lower when the number of agents in the system is larger. This can be seen with the difference of peaks in figures 5.20a and 5.20b with 10 and 30 agents.

Environment Modifications

Environment modifications affect agents' perceptions and their positions are modified. Formally, this results in a modification of some variables' domain and soft constraints. The first studied perturbation is a modification of the environment, without a modification of the fleet of CAV. So, only perceptions are changing without any position change. Then a simulation of a modification of the fleet is considered. This is concretely done by modifying variables and soft constraints, resulting in more soft constraints and variables changes.

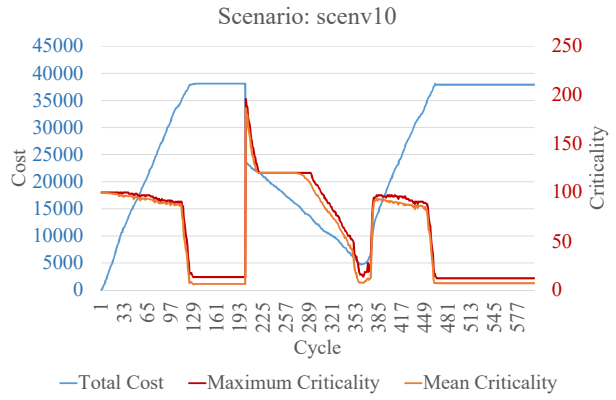


(a) CODCOP behaviour with an agent leaving the environment at the 200^{th} cycle with 10 initial agents.

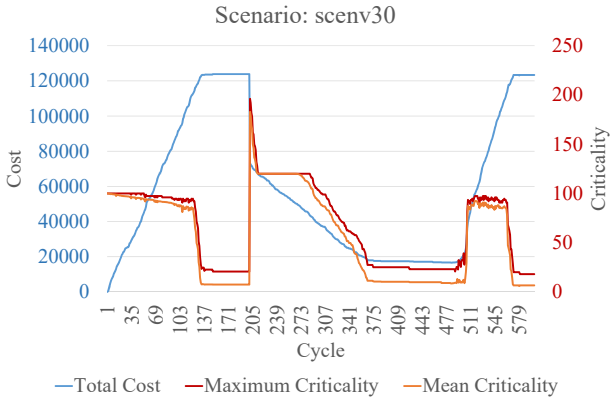


(b) CODCOP behaviour with an agent leaving the environment at the 200th cycle with 30 initial agents.

Figure 5.20: System evolution with an agent leaving the environment



(a) CODCOP behaviour with an environment modification at the 200th cycle with 10 initial agents.



(b) CODCOP behaviour with an environment modification at the 200th cycle with 10 initial agents.

Figure 5.21: System evolution with the environment evolving

Figures 5.21a and 5.21b show how the system adapts to a modification of the environment. As expected the criticality rises immediately after the changes. This criticality exceeds 100 because the positive criticality ($crit1$) is equal to 0 while the criticality related to having too much useless message ($crit3$) reaches the maximum possible. At first, the criticality related to the number of messages was greater than 0. Because the maximum criticality is negative, the first action taken by agents is to stop sharing some messages (they put many variables to 0 that were previously at 1). This behaviour continues until the threshold between too many useless messages and not enough messages is reached. Then the positive criticality takes once again the ascendant and the number of messages exchanged begins to grow again. This last phase is similar to the initial phase when the system tries to converge. These experiments show that the system is able to adapt to changes. However, the number of required cycles to adapt is greater than the number of required cycles for the openness characteristic. This number of cycles may be a limit in some environments that must be taken into account. One way to solve it would be to drop the preceding solution and begin from scratch once again, putting the system into a state where the total cost equal to 0 instead of a temporary solution where the total cost is greater than 0. This must be decided upstream for the application according to preferences.

Synthesis on Dynamic

The CODCOP module has been experimented with several potential system disruptions. For any encountered situations, a disruption first leads to an increase of the criticality in the system. Consequently, agents re-organise themselves to lower the maximum criticality. Thus, the system converges again on a new solution according to the environment modifications. While re-organising, a temporary solution is always available. This is not the most ideal one but is better than having no solution available. The **openness** challenge is checked but an improvement has to be made on radical modifications affecting the overall environment. Further investigations should consider how to fasten re-organisation when the environment is disturbed. A solution would be to provide agents with the capacity to negotiate to exchange two variables. However, doing so would ask to separate the too much useless criticality (criticality 3) from the too much criticality (criticality 2) and may induce more communication.

5.3.5 CODCOP Synthesis

In this section, an AMAS-based contribution to address a dynamic and asymmetric Distributed Constraint Optimisation Problem, called CODCOP, has been evaluated. The DCOP framework has been used to model a communication problem in a fleet of CAVs. This scope of application requires addressing the challenges of **dynamicity**, **openness** and **scalability** that are assessed through many experiments. First, a comparison with the state-of-the-art methods has been conducted, then, the two required challenges have been checked using another set of experiments.

As the addressed DCOP is an asymmetric one, in consequence, only algorithms that take this characteristic into account can be used. Because time and privacy are required there is no possibility of using a centralised solution. Algorithms used to compare are then local search algorithm based on the MGM one. The use of MGM and MGM2 highlights clearly the asymmetric nature of the problem, but they are not adapted to address it. Moreover, MCS- and GCA-MGM extensions are not suitable as their particularities are not efficient towards the problem characteristics.

The CODCOP solution proposed in this thesis is efficient for dynamic and high dimensional problems. Thousands of experiments have been conducted generating a low increase in computation time, especially with a proper computation distribution. Moreover, the system reacts quickly to perturbations either from the environment or their local perceptions. For these two reasons, the challenges of dynamicity and scalability are matched. Moreover, agents do not share personal information to reach a solution, so the **privacy** challenge is checked.

There is a strong limit to take into account. The AMAS has been designed to address this particular problem that can be formalised in DCOP. Even if an AMAS generic algorithm has been proposed, no generic implementation exists yet. It is important to generalise the AMAS algorithm used in CODCOP, to any well designed DCOP. With this implementation, a proper evaluation as a DCOP method should be made and an **accurate** and **generic** implementation would give the tools to classify it.

5.4 Contributions Conclusions

The thesis focuses on two main challenges, **cross-understanding** and **communication optimisation** that are directly related to vehicle driving efficiency. Cooperation inside a fleet of CAV has many advantages and is a way to compensate for sensors failures. The **cross-understanding** challenge can be modelled as the encounter of a Non-Cooperative Situation between CAVs. If communicated information does not provide any new knowledge for the receiving CAV then the communication is seen as useless. This can be caused by the use of a different referential frame between both CAVs. The second challenge is to **optimise communications** at maximum. Such a challenge has been modelled as a Distributed Constraint Optimisation Problem in the CODCOP model.

This thesis contributions have shown a way to address communication problems inside a fleet of CAVs. Two AMAS-based modules, LUDA and CODCOP have been designed to address communication issues in a fleet of CAVs that may prevent cooperative strategies to be used in a fleet of Connected and Autonomous Vehicles. The LUDA module objective is to transform the value of the information communicated by other CAVs in the fleet to make this information usable by the Autonomous Driving Process. The CODCOP module objective is to optimise the communication in a fleet of CAV. In this chapter, both modules LUDA and CODCOP have been tested and evaluated according to challenges of **dynamicity**, **openness**, **scalability** and **privacy**.

The use of a common referential is mandatory for a fleet of CAVs because communication is prone to be misunderstood or not understood at all, thus leading to invalidate cooperative strategies based on it. In this context, the LUDA module has been designed. The LUDA module has been compared to state-of-the-art Data Estimation techniques. With the joint simple behaviour of the three types of defined agents in the system, results show an improvement in the transformation of a data value into another referential frame. Related information is correctly selected and can be used instead of a missing one. With the redundancy of the same data from different referential frames, the LUDA module decreases the influence of the noise. It has been shown that the LUDA module reaches its full potential with 20 histories for the *morph agents*. Using a strategy of agent self-destruction to remove useless agents, the LUDA module lowers its impact on both resources and computation time.

Many cooperative strategies are possible thanks to the exchange of information between CAVs. When addressing the most complex situations many strategies can be used at the same time, to achieve different objectives (improving comfort, decreasing the travel time, reducing energy consumption, etc). Each one of them requires one or several specific and reliable information to be effective. Using an AMAS-based solution, each agent selects from its information the most useful ones for the overall system while avoiding overloading another agent. As a naturally distributed constraint optimisation problem, the solution has been compared to the state-of-the-art ADCOP algorithm. Because of the scalability requirement, only matching algorithms are used for the comparison. The results show that the CODCOP solution is efficient to optimise communication in a fleet of CAVs, with regards to the challenge of **dynamicity**, **openness**, **scalability** and **privacy**. However, these results must be tempered because the CODCOP has been specifically designed to address this problem. The fact that the common assumption of only one variable by an agent is not matched is a clear limitation to the comparison. In addition, the system's characteristics have been explored according to the challenges of **dynamicity** and **scalability**. The increase of dimension has only a small impact on the computation time which assess the characteristic of **scalability**. Confronted to perturbations, the overall system is first impacted but quickly recover a stable state with no hard constraints violated which assess the characteristic of **dynamicity** and **openness**.

Chapter 6

Conclusions and Perspectives

The objective of this chapter is to conclude on these thesis contributions for the Intelligent Transport domain and the Multi-Agent System approach. Some future perspectives are then given on improvements and use of both modules presented in this work.

6.1 General Conclusion

To summarise in one sentence, this thesis answer the question: **"how to communicate efficiently in a fleet of CAVs"**. This is motivated by the multiple benefits that the exchange of information within a fleet of CAVs may bring. Indeed, in the state-of-the-art, there is a huge amount of efficient strategies, systems and architectures for a fleet of CAVs using communication as a basis. With all these strategies, come the multiple benefits of pollution and travel time reduction, a better traffic flow, fewer traffic jams with improvement for security, health and comfort but also in an economical way. This implies that having reliable communication is mandatory. Even if, many causes may disrupt communication. This thesis contributions have focused on designing two modules to tackle the challenges of **cross-understanding** and **communication optimisation** in a fleet of CAVs. Then, the first addressed problem can be stated as:

How to understand an information received from another referential frame?

It has been shown that it is possible to adapt an information from a referential frame into another and make it usable by estimating its value. In consequence, the information becomes useful and then the overall quality of communication is increased. To cope with the problems of **dynamics, openness and scalability** that characterise the road traffic context, the proposed solution, called LUDA, is based on the AMAS approach. With the definition of three types of agents (*morph agents*, *data agents* and *group agents*), the LUDA module enables a CAV to transform a received information, and give a value to the Autonomous Driving Process (ADP) of the CAV that uses it to make a decision. In this proposition, each agent has a local objective and when all agents have reached a satisfying state that copes with their objective the overall system is functionally adequate. All three types of agents cooperate to obtain an organisation enabling to receive an information and transform it. A *morph agent* transforms the received information into the correct referential frame. Then, the corresponding *data agent* receives the transformed value and gives it its *group agent*. Finally, a *group agent* aggregates all values representing the same environmental information into a more accurate one to give it to the ADP.

To obtain this organisation, first, a*data agents* are created when the CAV perceives a new environmental information. This information can be perceived by the sensors or received thanks to the communication of another CAV. One environmental information can be perceived and received in several forms (depending on the referential frame, for example, kilometres and miles). So, two *data agents* may represent the same environmental information. This is why they have the objective of "finding the other *data agents* related to the same environmental information but expressed in another referential frame". Because a *data agent* has no information about the other *data agents* related to the same environmental information, it creates a *morph agent* for each *data agent*, to find if they are related. A *morph agent's* objective is to adapt the value from a *data agent* (its creator) information into the value of another *data agent* (its objective) using a transformation function. To obtain this function, the *morph agent* behaviour is based on a linear regression method. If the *morph agent* find a transformation function that can estimate from its *data agent* creator's value the value of the *data agent* objective, it means that the two *data agents* are related to the same environmental information. In consequence, they group themselves thanks to a *group agent*. Intending to have the most reliable value to give to the ADP (information can be noisy), a *group agent* has two sub-goals to address this issue: 1) to group only related *data agents* and 2) to combine efficiently the value of each grouped *data agents* to improve the value it gives to the ADP. To summarise, if environmental information is missing for a CAV to make a decision but another CAV communicates it using another referential frame, the following process is possible: 1) the corresponding *morph agent* transform the received value, 2) the corresponding *data agent* gives this value to the *group agent* and 3) the *group agent* uses this value to give it as input for the ADP. When several *data agents* in the same group gives the value to the *group agent*, it combines all values into one.

The LUDA module has been evaluated with regard to the state-of-the-art techniques of data estimation, presenting better results in a large-scale, dynamic and open context. Experiments exploring the behaviour of the LUDA module confronted with different scales and noises assess the effectiveness of the module for these characteristics.

Thanks to the LUDA module, all information can be understood within a fleet of CAVs. However, it remains a question to address:

How to ensure that all the shared information are useful?

Indeed, a CAV perceives a huge amount of information continuously. Adopting a strategy where CAVs exchange the totality of their perception is not suitable for large-scale environments. A CAV may be in a situation where the amount of information to process is so huge that the time required to process them exceeds the time available to make a decision. Rather than being a help, communication becomes a problem. In this thesis, a cooperative strategy has been proposed to avoid reaching this limit while selecting the most useful information to share. Still, with the characteristics linked to road traffic (**dynamicity**, **openness** and **scalability**), the problem of **privacy** is added with the question:

How to share preferences without sharing private information?

This question highlights the concern of exchanging private information. Such an information is sensitive and many rules are constructed to limit their use (for example, the General Data Protection Regulation in the European Union). To cope with this constraint, an AMAS-based module, called CODCOP, has been designed to allow a CAV to **optimise its communications** without overloading any CAVs of the fleet. It uses the formalism of the well-known framework Distributed Constraint Optimisation (DCOP). To do so, the AMAS concept of criticality to exchange an idea of their preferences instead of the full private information is used. From the set of all received information, a CAV can evaluate if the communication volume can be processed and if the received information is useful to the CAV. This is achieved with three different criticalities: 1)

the estimation of missing useful information compared to the maximum processing capacity of the CAV; 2) the volume of messages compared to the maximum processing capacity of the CAV; 3) the volume of messages not providing new knowledge compared to the maximum processing capacity of the CAV. All three criticalities are aggregated into one and shared with other CAVs in the neighbourhood. By doing so, agents can understand the information to share to lower the criticalities, and it results in the optimisation of the overall communication quality without sharing private information.

The proposed contribution, called CODCOP, is a local search, near-optimal and incomplete solution making it suitable for large-scale problems. Presented experiments explore how the module behaves with differents scales but also with different perturbations. Results show a great adaptation capacity where agents understand that a re-organisation is necessary and act according to it. The **openness** characteristic is addressed by the local behaviour of every agent. However, the resulting **dynamic** changes take more time for the system to converge again to a stable state. The computation time increases slowly with the augmentation of dimension making it suitable for the characteristic of **scalability**.

With these two modules added to the ADP of a CAV, it ensures that cooperative strategies based on the CAVs exchange of information are not disrupted because they cannot understand each other or because the overall amount of information prevents it to be processed.

6.2 Contribution to Intelligent Transport System

Intelligent Transport Systems (ITS) is a subset to the smart city domain, that aims to improve the quality of life for all users. In this context, road users are mainly focused but other domains can also benefit from the improvement. Three main objectives are addressed by ITS: travel time, energy consumption and pollution. Numerous studies have been proposed and designed during the last three decades. Among the works available in the scientific literature, the fleet of CAVs stands out as a solution to improve these three objectives. Indeed, by using an autonomous driving process in addition to communication, CAVs can share a common vision of their environment, sharing a useful information or negotiating the best way to address a situation. With these possibilities, CAVs have access to strategies that are expected to improve the users' quality of life.

This thesis contribution on ITS is related to the communication inside a fleet of CAVs. Many works of the state-of-the-art base their solution on the effective and reliable exchange of information between CAVs. However, state-of-the-art works require that communication is reliable and understandable to be efficient. Such assumptions can strongly limit the capacity of a CAV to address any situation it encounters. Indeed, three causes may prevent the correct understanding of information: 1) the lack of a common referential frame; 2) the use of different units of measurement for the same information; 3) the impossibility for the CAV to process the information in time. In consequence, there is a need give to the CAV the tools to understand communication from in another referential frame than its own and to select the best information to share for any situations without overloading other CAVs. This thesis proposes a generic way to address these needs, without the use of semantics, thanks to the two different modules, LUDA and CODCOP, that have been designed to answer the ITS application requirements that are dynamic, openness, privacy and scalability.

6.3 Contribution on the Multi-Agent Systems Field

The Multi-Agent System approach emphasises the local behaviour of entities called "agents" and it is efficient to address complex systems. It is also highly efficient for dynamic and large problems. Moreover, focusing on local behaviours enables the agents to keep private information only for their owners. For all these reasons, the Multi-Agent System approach has been chosen for addressing this thesis' challenges. Moreover, the Adaptive Multi-Agent System (AMAS) approach enables to find a solution to incompletely specified problems.

In this thesis, two AMAS have been designed to tackle two problems. These problems are known issues and challenges in the AMAS approach. Usually, communication is assumed to be reliable and understandable. When this assumption is not checked two non-cooperative situations (NCS) are encountered leading the overall system to not be functional. These two NCS are: 1) **incomprehension** when the communication structure is unknown for the agent (for example, an information using another referential frame) and 2) **unproductiveness** when no new knowledge is extracted from the communication (for example two instances of the same information). In consequence, it is mandatory either to ensure that these assumptions are matched or to give the agents a solution to get out of such NCSs. In this thesis, an answer to the NCS of incomprehension has been given. Because agents are cooperative, an agent receiving an information assumes that the received information must be useful. Any information is useful if it provides new knowledge for the receiving agent. In consequence, the agent transforms the received information to create knowledge it is missing. Thanks to several examples, the agent can transform the information into a useful one. Thus the incomprehension NCS is no longer encountered.

The second NCS tackled in this thesis is the **unproductiveness** one. The amount of exchanged information between agents can exceed the computation capabilities of agents. In that sense, an agent is given the tools to select the information useful to communicate to the proper functioning and avoid the ones that do not help its neighbours.

6.4 Perspectives

This thesis proposes two contributions that are relevant for the Intelligent Transport Systems (ITS) as context. From a theoretical view, both modules can be improved to increase their results. The LUDA module organisation may be accelerated by several cooperative strategies. But, the LUDA module can be a base for future works. The question of sharing the acquired knowledge can also be asked. As cooperative entities, it is possible that experimented CAVs share their knowledge with others. This involves two aspects that are still difficult nowadays: sharing the knowledge **fast** and **accurately**. Recent works present solutions to transfer the knowledge from one system to another, or from one referential frame to another one to be used for different applications. For example, object detection in an image with several backgrounds.

The second module, CODCOP, provides a solution to a problem where an agent cannot reach its goal. The agent requires the help of its neighbours to increase its knowledge about the environment. Because neighbours agents are cooperative, they share information they think is useful, and thus the agent has more information to make a decision. In such a system, an agent needs the others but without exchanging their personal information. With the increased need for privacy but also the growth of IoT, it is important to enable many devices to work together while keeping information private. It would be interesting to use the CODCOP module in an IoT context to evaluate the benefits on the optimisation with a more static environment.

However, it has not been possible to experiment with the proposed modules on road traffic scenarios. This is because no road traffic simulator provides all wanted characteristics currently. However, many can include them with some improvements. In the future, the LUDA and CODCOP modules will be added to the ADP of CAVs in a road traffic simulator. For the LUDA module, this simulator must ensure four characteristics:

- It simulates the micro-behaviour of a vehicle and has an ADP making all decisions. The ADP makes its decision using a set of information that is accessible to be modified.

- The simulator must simulate the real-world complexity with every road user but also the environmental conditions (weather, non-autonomous vehicles, pedestrian, etc). In addition, many unpredictable events must be included (for example, a leaf blocking a sensor).

- Real sensors must be simulated with all consequences that imply. Information is not always perfect, and an obstacle prevents some information to be obtained in its totality.
- The use of real sensors implies the use of algorithms detecting in simulation-time the perceptions of the CAV, and not access to the simulated value directly.

Thanks to these four characteristics it is possible to construct an accurate experiment for the LUDA module with the following steps. First, a cooperative strategy from the state-of-the-art is implemented in the simulator with perfect information. Then, information is degraded with noise and each CAV uses a local referential frame. Because of this, the cooperative strategy is expected to be impossible to follow (communication not understood and/or not reliable). Then, the ADP is equipped with LUDA module and the cooperative strategy can once again be followed. The CODCOP module can also be tested with a road traffic simulator with similar characteristics. In this case, the objective is to add a high number of CAVs, that gather a lot of information. With the use of the CODCOP module, the communication volume will remain at a low level and only the most useful information would be passed on. At the time this thesis is written, the implementation of a simulation with the stated characteristics has begun, it is based on the Carla simulator [32].

From such a simulation, it would be possible to make complementary experiments in the medium and long term, enabling to evaluate:

- The benefits of cooperation on the CAVs' behaviour.
- The requirement in terms of time to communicate an information.
- The potential issue caused by malevolent entities in the environments, which may cause, in the worst case, accidents.
- How a CAV can be accepted by road users.

ACKNOWLEDGEMENTS

This work is part of the neOCampus operation of the Université Toulouse III Paul Sabatier. www.neocampus.org.

Glossary

Concept	Abbreviation	Definition
Smart City	-	The smart city is a concept of urban development. It is about improving the quality of life of city citizens by making the city more adaptive and efficient, using new technologies that rely on an ecosystem of objects and services.
Intelligent Transport System	ITS	Intelligent transport systems are the applications of new information and communication technologies to transport and its logistics.
Autonomous Vehicle	AV	A vehicle that can drive without human control. Also called self-driving car.
Connected Vehicle	CV	A vehicle connected to the network
CAV	CAV	A vehicle connected to the network and that can drive without human control
Fleet of CAV	-	Several CAVs that form a coalition to improve the overall traffic state
Platoon	-	A particular form of fleet where all CAVs are connected together with a common goal. The front CAV is known as the leader, making all the platoon's decisions
Road-Side Unit	RSU	A static intelligent device situated close to the road.
On-Board Unit	OBU	Embedded in a CAV, it is in charge of reading data from on-board sensors, determining which manoeuvres to initiate, and exchanging sensors data with neighbouring vehicles
Autonomous Driving Process	ADP	The system responsible of taking all decision for an autonomous vehicle
Vehicle-to-Vehicle	V2V	Communication taking place between two CVs
Vehicle-to-Infrastructure	V2I	Communication taking place between a CV and a RSU
Infrastructure-to-Infrastructure	I2I	Communication taking place between two RSUs
Vehicle-to-everything	V2X	Communication taking place between two non-specific entities
Internet of Thing	IoT	The interconnection of many intelligent devices
Internet of Vehicles	IoV	The interconnection of many intelligent vehicles
Mobile Ad-Hoc Network	MANET	Is a wireless, multi-hop, self-configuring network used in the IoT domain
Vehicular Ad-Hoc Network	VANET	Is a special class extended from MANET with predefined routes
History	-	A collection of past observations

Table 6.1: Proposition of a glossary.

Own Bibliography

[1] Davide Andrea Guastella, Guilhem Marcillaud, and Cesare Valenti. Edge-based missing data imputation in large-scale environments. *Information*, 12(5):195, 2021.

[2] Guilhem Marcillaud, Valérie Camps, Stéphanie Combettes, Marie-Pierre Gleizes, and Elsy Kaddoum. Management of intelligent vehicles: Comparison and analysis. In *12th International Conference on Agents and Artificial Intelligence (ICAART 2020)*, pages 258–265, 2020.

[3] Guilhem Marcillaud, Valérie Camps, Stéphanie Combettes, Marie-Pierre Gleizes, and Elsy Kaddoum. A self-adaptive module for cross-understanding in heterogeneous multiagent systems. In *ICAART (1)*, pages 353–360, 2021.

[4] Filipo Studzinski Perotto, Stéphanie Combettes, Valérie Camps, Fabien Michel, Kristell Aguilar Alarcon, Loïc Caroux, Maxime Delmas, Marie-Pierre Gleizes, Elsy Kaddoum, Elodie Labeye, et al. Projet c2c–véhicules connectés et acceptabilité par l'humain. In *Les Journées Francophones de la Modélisation et de la Simulation (JFMS 2020)*. Cépaduès, Toulouse, 2020.

[5] Filipo Studzinski Perotto, Stéphanie Combettes, Valérie CAMPS, Elsy Kaddoum, Guilhem Marcillaud, Pierre Glize, and Marie-Pierre Gleizes. Integrating Shared Information into the Sensorial Mapping of Connected and Autonomous Vehicles. In *13th International Conference on Agents and Artificial Intelligence (ICAART 2021)*, volume 1 of *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, pages 454–461, Lisbonne (virtuel), Portugal, February 2021. SciTePress.

Bibliography

[6] A. Agarwal and P. Paruchuri. V2v communication for analysis of lane level dynamics for better EV traversal. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 368–375, 2016.

[7] Kristell Aguilar Alarcon, Marie-Pierre Gleizes, Loïc Caroux, Elsy Kaddoum, Valérie CAMPS, and Stéphanie Combettes. ReCoVac : Décision autonome de reprise de la conduite d'un véhicule autonome et connecté. In *JFSMA 2020, Systèmes Multi-Agents : Architectures Multi-Agents pour la Simulation de Systèmes Complexes*, Angers [en ligne], France, June 2020.

[8] Alessandro Aliberti, Lorenzo Bottaccioli, Enrico Macii, Santa Di Cataldo, Andrea Acquaviva, and Edoardo Patti. A non-linear autoregressive model for indoor air-temperature predictions in smart buildings. *Electronics*, 8(9):979, 2019.

[9] Salim Allal and Saadi Boudjit. Geocast routing protocols for vanets: Survey and guidelines. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 323–328. IEEE, 2012.

[10] M. H. Almannaa, Hao Chen, Hesham A. Rakha, Amara Loulizi, and Ihab El-Shawarby. Field implementation and testing of an automated eco-cooperative adaptive cruise control system in the vicinity of signalized intersections. *Transportation Research Part D: Transport and Environment*, 67:244–262, 2019.

[11] Omar Bagdadi and András Várhelyi. Jerky driving—an indicator of accident proneness? *Accident Analysis and Prevention*, 43(4):1359–1363, 2011.

[12] Hamidreza Bagheri, Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Dirk Pesch, Klaus Moessner, and Pei Xiao. 5g nr-v2x: Toward connected and cooperative autonomous driving. *IEEE Communications Standards Magazine*, 5(1):48–54, 2021.

[13] Tudor Barbu. Variational Image Denoising Approach with Diffusion Porous Media Flow. In *Abstract and Applied Analysis*, volume 2013, pages 1 – 8. Hindawi, 2013.

[14] J. Barthelemy and T. Carletti. A dynamic behavioural traffic assignment model with strategic agents. *Transportation Research Part C: Emerging Technologies*, 85:23–46, 2017.

[15] Ana LC Bazzan and Franziska Klügl. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(3):375–403, 2014.

[16] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo – simulation of urban mobility: An overview. In SINTEF & University of Oslo Aida Omerovic, RTI International Research Triangle Park Diglio A. Simoni, and RTI International Research Triangle Park Georgiy Bobashev, editors, *SIMUL 2011*. ThinkMind, 2011.

[17] Clara Benevolo, Renata Paola Dameri, and Beatrice D'auria. Smart mobility in smart city. In *Empowering organizations*, pages 13–28. Springer, 2016.

[18] Jérémy Boes. *Apprentissage du contrôle de systèmes complexes par l'auto-organisation coopérative d'un système multi-agent: application à la calibration de moteurs à combustion*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2014.

[19] Noelie Bonjean, Wafa Mefteh, Marie-Pierre Gleizes, Christine Maurel, and Frédéric Migeon. Adelfe 2.0. In *Handbook on agent-oriented design processes*, pages 19–63. Springer, 2014.

[20] Jonathan Bonnet. *Multi-criteria and multi-objective dynamic planning by self-adaptive multi-agent system, application to earth observation satellite constellations*. PhD thesis, Université Paul Sabatier-Toulouse III, 2017.

[21] Rasa Bruzgiene, Lina Narbutaite, and Tomas Adomkus. Manet network in internet of things system. *Ad Hoc Networks*, pages 89–114, 2017.

[22] Kelvin Joseph Bwalya. Spatial intelligence and vehicle-to-vehicle communication: Topologies and architectures. In *Connected Vehicles in the Internet of Things*, pages 19–36. Springer, 2020.

[23] Ziyu Chen, Chen He, Zhen He, and Minyou Chen. Bd-adopt: a hybrid dcop algorithm with best-first and depth-first search strategies. *Artificial Intelligence Review*, 50(2):161–199, 2018.

[24] Liel Cohen and Roie Zivan. Max-sum revisited: The real power of damping. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 111–124. Springer, 2017.

[25] Stefano Coniglio, Fabio Furini, and Pablo San Segundo. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research*, 289(2):435–455, 2021.

[26] Wendell Cox. Demographia world urban areas: 17th annual edition. *Demographia*, 2021.

[27] HSM Coxeter. The mathematics of map coloring. In *Mathematical Solitaires & Games*, pages 135–144. Routledge, 2019.

[28] Alaa Daoud, Flavien Balbo, Paolo Gianessi, and Gauthier Picard. A generic agent model towards comparing resource allocation approaches to on-demand transport with autonomous vehicles. In *OptLearnMAS-21: The 12th Workshop on Optimization and Learning in Multiagent Systems*, 2021.

[29] Bruno Dato. *Lifelong Learning by Endogenous Feedback Application to a Robotic System*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2021.

[30] Rina Dechter, David Cohen, et al. *Constraint processing*. Morgan Kaufmann, 2003.

[31] Augustin Degas. *Auto-structuration de trafic temps-réel multi-objectif et multi-critère dans un monde virtuel*. PhD thesis, Université Paul Sabatier - Toulouse III, 2020.

[32] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[33] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

[34] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff. The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. *IEEE Wireless Communications*, 23(4):146–152, 2016.

[35] Tanguy Esteoule. *Adaptive Multi-Agent Systems for Wind Power Forecasting*. PhD thesis, Université Paul Sabatier-Toulouse III, 2019.

[36] Bei Fang, Ying Li, Haokui Zhang, and Jonathan Cheung-Wai Chan. Hyperspectral images classification based on dense convolutional networks with spectral-wise attention mechanism. *Remote Sensing*, 11(2), 2019.

[37] Jacques Ferber and Gerhard Weiss. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999.

[38] Juan Angel Ferreiro-Lage, Cristina Pereiro Gestoso, Oscar Rubiños, and Fernando Aguado Agelet. Analysis of unicast routing protocols for vanets. In *Fifth International Conference on Networking and Services*, pages 518–521, 2009.

[39] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018.

[40] FIA Foundation. Make road safe, 2020.

[41] Jeroen Fransman, Joris Sijs, Henry Dol, Erik Theunissen, and Bart De Schutter. Bayesian-dpop for continuous distributed constraint optimization problems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1961–1963, 2019.

[42] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[43] French-government. Key figures for french road traffic. https://www.statistiques.developpement-durable.gouv.fr/, 2019. Accessed: 2019-02-18.

[44] M. Gaciarz, Samir Aknine, and Neila Bhouri. Automated negotiation for traffic regulation. *Advances in Social Computing and Multiagent Systems*, pages 1–18, 2015.

[45] Jean-Pierre Georgé, Marie-Pierre Gleizes, and Pierre Glize. Conception de systèmes adaptatifs à fonctionnalité émergente: la théorie amas. *Revue d'intelligence artificielle*, 17(4):591–626, 2003.

[46] Kate Gibson. Arizona police: Pedestrian stepped in front of self-driving uber before crash, 2018.

[47] Marie-Pierre Gleizes, Valérie Camps, Anthony Karageorgos, and Giovanna Di Marzo Serugendo. Agents and multi-agent systems. In *Self-organising Software*, pages 105–119. Springer, 2011.

[48] Pierre Glize. L'adaptation des systèmes à fonctionnalité émergente par auto-organisation coopérative. *Hdr, Université Paul Sabatier, Toulouse III*, 2001.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[50] Tal Grinshpoun. When you say (dcop) privacy, what do you mean?-categorization of dcop privacy and insights on internal constraint privacy. In *International Conference on Agents and Artificial Intelligence*, volume 2, pages 380–386. SCITEPRESS, 2012.

[51] Tal Grinshpoun, Alon Grubshtein, Roie Zivan, Arnon Netzer, and Amnon Meisels. Asymmetric distributed constraint optimization problems. *Journal of Artificial Intelligence Research*, 47:613–647, 2013.

[52] Tal Grinshpoun and Tamir Tassa. P-syncbb: A privacy preserving branch and bound dcop algorithm. *Journal of Artificial Intelligence Research*, 57:621–660, 2016.

[53] Davide Andrea Guastella, Valérie Camps, and Marie-Pierre Gleizes. A cooperative multi-agent system for crowd sensing based estimation in smart cities. *IEEE Access*, 8:183051–183070, 2020.

[54] M. Gueriau, Romain Billot, Nour-Eddin El Faouzi, Julien Monteil, Frederic Armetta, and Salima Hassas. How to assess the benefits of connected vehicles? a simulation framework for the design of cooperative traffic management strategies. *Transportation Research Part C: Emerging Technologies*, 67:266–279, 2016.

[55] Leonardo Guevara and Fernando Auat Cheein. The role of 5g technologies: Challenges in smart cities and intelligent transportation systems. *Sustainability*, 12(16):6469, 2020.

[56] Umar Zakir Abdul Hamid, Fakhrul Razi Ahmad Zakuan, Khairul Akmal Zulkepli, Muhammad Zulfaqar Azmi, Hairi Zamzuri, Mohd Azizi Abdul Rahman, and Muhammad Aizzat Zakaria. Autonomous emergency braking system with potential field risk assessment for frontal collision mitigation. In *2017 ieee conference on systems, process and control (icspc)*, pages 71–76. IEEE, 2017.

[57] Kyungtae Han, BaekGyu Kim, Prashant Tiwari, Ziran Wang, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J. Barth. Cooperative ramp merging system: Agent-based modeling and simulation using game engine. *SAE International Journal of Connected and Automated Vehicles*, 2(2):115–128, may 2019.

[58] David Hasenfratz, Olga Saukh, Christoph Walser, Christoph Hueglin, Martin Fierz, Tabita Arn, Jan Beutel, and Lothar Thiele. Deriving high-resolution urban air pollution maps using mobile sensor nodes. *Pervasive and Mobile Computing*, 16:268–285, 2015.

[59] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Vanet security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.

[60] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *International conference on principles and practice of constraint programming*, pages 222–236. Springer, 1997.

[61] Katsutoshi Hirayama and Makoto Yokoo. The distributed breakout algorithms. *Artificial Intelligence*, 161(1-2):89–115, 2005.

[62] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[63] Khoi Hoang, Ferdinando Fioretto, Ping Hou, Makoto Yokoo, William Yeoh, and Roie Zivan. Proactive dynamic distributed constraint optimization. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2016.

[64] Khoi D Hoang, Ping Hou, Ferdinando Fioretto, William Yeoh, Roie Zivan, and Makoto Yokoo. Infinite-horizon proactive dynamic dcops. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 212–220, 2017.

[65] John H Holland. *Complexity: A very short introduction*. OUP Oxford, 2014.

[66] Tongyi Huang, Wu Yang, Jun Wu, Jin Ma, Xiaofei Zhang, and Daoyin Zhang. A survey on green 6g network: Architecture and technologies. *IEEE Access*, 7:175758–175768, 2019.

[67] INRIX. Embouteillages : Une facture cumulee de plus de 350 milliards d'euros pour la france sur les 16 prochaines annees, 2013.

[68] INRIX. Congestion costs each american nearly 100 hours, $1,400 a year, 2020.

[69] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth. Multi-agent intersection management for connected vehicles using an optimal scheduling approach. In *2012 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 185–190, 2012.

[70] Tom Jorquera. *An adaptive multi-agent system for self-organizing continuous optimization*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2013.

[71] Susanne Kalenka and Nicholas R Jennings. Socially responsible decision making by autonomous agents. In *Cognition, Agency and Rationality*, pages 135–149. Springer, 1999.

[72] M. A. S. Kamal, J. Imura, T. Hayakawa, A. Ohata, and K. Aihara. A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1136–1147, 2015.

[73] M. A. S. Kamal, S. Taguchi, and T. Yoshimura. Efficient vehicle driving on multi-lane roads using model predictive control under a connected vehicle environment. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 736–741, 2015.

[74] Frédéric Kaplan. L'émergence d'un lexique dans une population d'agents autonomes. *These de l'université Paris VI*, 2000.

[75] Judy Kegl. Creation through contact: Sign language emergence and sign language change in nicaragua. *Language creation and language change: Creolization, diachrony, and development*, 1999.

[76] Grace Khayat, Constandinos X Mavromoustakis, George Mastorakis, Hoda Maalouf, Jordi Mongay Batalla, Evangelos Pallis, and Evangelos K Markakis. Intelligent vehicular networking protocols. In *Convergence of Artificial Intelligence and the Internet of Things*, pages 59–86. Springer, 2020.

[77] Tai-hoon Kim, Carlos Ramos, and Sabah Mohammed. Smart city and iot. *Future Generation Computer Systems*, 76:159–162, 2017.

[78] J. Kovalevsky and Ivan I. Mueller. *Reference coordinate systems for Earth Dynamics*, volume 154, pages 1–12. Springer Netherlands, Dordrecht, 1989.

[79] Kranti Kumar, M Parida, and VK Katiyar. Short term traffic flow prediction for a non urban highway using artificial neural network. *Procedia-Social and Behavioral Sciences*, 104:755–764, 2013.

[80] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193–207, 1990.

[81] Kuo-Yun Liang, Jonas Mårtensson, and Karl H Johansson. Heavy-duty vehicle platoon formation for fuel efficiency. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1051–1061, 2015.

[82] S.-H. Lin and T.-Y. Ho. Autonomous vehicle routing in multiple intersections. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 585–590. ACM, 2019.

[83] Yangxin Lin, Ping Wang, and Meng Ma. Intelligent transportation system (its): Concept, challenge and opportunity. In *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*, pages 167–172. IEEE, 2017.

[84] I. Llatser, S. KÃ¼hlmorgen, A. Festag, and G. Fettweis. Greedy algorithms for information dissemination within groups of autonomous vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[85] Seng W Loke. Cooperative automated vehicles: a review of opportunities and challenges in socially intelligent vehicles beyond networking. *IEEE Transactions on Intelligent Vehicles*, 4(4):509–518, 2019.

[86] Rajiv T Maheswaran, Jonathan P Pearce, and Milind Tambe. Distributed algorithms for dcop: A graphical-game-based approach. In *ISCA PDCS*, pages 432–439, 2004.

[87] Zaigham Mahmood. Connected vehicles in the internet of things: Concepts, technologies and frameworks for the IoV. In Zaigham Mahmood, editor, *Connected Vehicles in the Internet of Things*. Springer International Publishing, 2020.

[88] Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pages 438–445. IEEE, 2004.

[89] Shlomi Maliah, Guy Shani, and Roni Stern. Collaborative privacy preserving multi-agent planning. *Autonomous Agents and Multi-Agent Systems*, 31(3):493–530, 2017.

[90] S Manu and P Sivraj. Performance comparison of communication technologies for v2x applications. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 356–362. IEEE, 2020.

[91] J González Martín-Moro, F Prieto Garrido, F Gómez Sanz, I Fuentes Vega, M Castro Rebollo, and P Moreno Martín. Which are the colors of the dress? review of an atypical optic illusion. *Archivos de La Sociedad Española de Oftalmología (English Edition)*, 93(4):186–192, 2018.

[92] Tshilidzi Marwala. *Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques: Knowledge Optimization Techniques*. IGI Global, 2009.

[93] Barbara M Masini, Cristiano M Silva, and Ali Balador. Advances in vehicular networks. *Journal of Sensor and Actuator Networks*, 2020.

[94] Toshihiro Matsui, Hiroshi Matsuo, Marius-Calin Silaghi, Katsutoshi Hirayama, Makoto Yokoo, Satomi Baba, et al. A quantified distributed constraint optimization problem. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1023–1030, 2010.

[95] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[96] Wafa Mefteh. *Simulation Based Design For Adaptive Multi-Agent Systems with the ADELFE Methodology*. PhD thesis, Université Toulouse III Paul Sabatier, 2014.

[97] Wafa Mefteh, Frederic Migeon, Marie-Pierre Gleizes, and Faiez Gargouri. Adelfe 3.0 design, building adaptive multi agent systems based on simulation a case study. In Manuel Núñez, Ngoc Thanh Nguyen, David Camacho, and Bogdan Trawiński, editors, *Computational Collective Intelligence*, pages 19–28, Cham, 2015. Springer International Publishing.

[98] MIT. Moral machine platform, 2021.

[99] Michael Molloy, Michael S Molloy, and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23. Springer Science & Business Media, 2002.

[100] Noé Monsaingeon, Loïc Caroux, Sabine Langlois, Yovan Hurgobin, and Céline Lemercier. Driver compliance with automation reliability information regarding hazardous environmental circumstances. *Le travail humain*, 83(4):343–360, 2020.

[101] J. Monteil, Romain Billot, Jacques Sau, Frederic Armetta, Salima Hassas, and Nour Eddin El Faouzi. Cooperative highway traffic: multi-agent modelling and robustness to local perturbations. *Transportation Research Board 92nd Annual Meeting*, page 25, 2013.

[102] Samuel K Moore. Super-accurate gps coming to smartphones in 2018 [news]. *IEEE Spectrum*, 54(11):10–11, 2017.

[103] A. Mosebach, S. Röchner, and J. Lunze. Merging control of cooperative vehicles. *IFAC-PapersOnLine*, 49(11):168–174, 2016.

[104] E. Mouhcine, Khalifa Mansouri, and Youssfi Mohamed. Solving traffic routing system using VANet strategy combined with a distributed swarm intelligence optimization. *Journal of Computer Science*, page 14, 2018.

[105] Florent Mouysset, Célia Picard, Christophe Bortolaso, Frederic Migeon, Marie-Pierre Gleizes, Christine Maurel, and Mustapha Derras. Investigations of process mining methods to discover process models on a large public administration software. In *37ème Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID 2019)*, pages 147–162, 2019.

[106] Anthony J Myles, Robert N Feudale, Yang Liu, Nathaniel A Woody, and Steven D Brown. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004.

[107] Mostofa Kamal Nasir, Rafidah Md Noor, M. A. Kalam, and B. M. Masum. Reduction of fuel consumption and exhaust pollutant using intelligent transport systems. *The Scientific World Journal*, 2014:836375, 2014.

[108] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. Distributed gibbs: A memory-bounded sampling-based dcop algorithm. *AAMAS '13*, page 167–174, 2013.

[109] James J Odell, H Van Dyke Parunak, Mitch Fleischer, and Sven Brueckner. Modeling agents and their environment. In *International Workshop on Agent-Oriented Software Engineering*, pages 16–31. Springer, 2002.

[110] Mark JL Orr et al. Introduction to radial basis function networks, 1996.

[111] Jesús Hamilton Ortiz and Alvaro Pachon De La Cruz. *Ad Hoc Networks*. BoD – Books on Demand, 2017. Google-Books-ID: kPSODwAAQBAJ.

[112] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. Duct: An upper confidence bound approach to distributed constraint optimization problems. *Association for Computing Machinery*, 8(5), 2017.

[113] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[114] Adrian Petcu and Boi Faltings. Mb-dpop: A new memory-bounded algorithm for distributed optimization. In *International Joint Conference on Artificial Intelligence*, pages 1452–1457, 2007.

[115] Adrian Petcu, Boi Faltings, and Roger Mailler. Pc-dpop: A new partial centralization algorithm for distributed optimization. In *International Joint Conference on Artificial Intelligence*, volume 7, pages 167–172, 2007.

[116] Gauthier Picard and Marie-Pierre Gleizes. The adelfe methodology. In *Methodologies and Software Engineering for Agent Systems*, pages 157–175. Springer, 2004.

[117] George Polya. *How to solve it: A new aspect of mathematical method*, volume 85. Princeton university press, 2004.

[118] N. Pulter, Heiko Schepperle, and Klemens Böhm. How agents can help curbing fuel combustion: A performance study of intersection control for fuel-operated vehicles. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Volume 2*, pages 795–802, 2011.

[119] Luo Qi. Research on intelligent transportation system technologies and applications. In *2008 Workshop on Power Electronics and Intelligent Transportation System*, pages 529–531. IEEE, 2008.

[120] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[121] Christian Quadri, Vincenzo Mancuso, Marco Ajmone Marsan, and Gian Paolo Rossi. Platooning on the edge. In *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 1–10, 2020.

[122] Inês FG Reis, Ivo Gonçalves, Marta AR Lopes, and Carlos Henggeler Antunes. A multi-agent system approach to exploit demand-side flexibility in an energy community. *Utilities Policy*, 67:101114, 2020.

[123] J. Rios-Torres and A. A. Malikopoulos. Automated and Cooperative Vehicle Merging at Highway On-Ramps. *IEEE Transactions on Intelligent Transportation Systems*, 18(4), 2017.

[124] Alex Rogers, Alessandro Farinelli, Ruben Stranders, and Nicholas R Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

[125] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[126] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[127] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[128] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[129] Pierre Rust. *Autonomous and Spontaneous Coordination between Smart Connected Objects Coordination spontanée et autonome entre objets intelligents connectés*. PhD thesis, Université de Lyon, 2019.

[130] Pierre Rust, Gauthier Picard, and Fano Ramparany. pydcop, a dcop library for iot and dynamic systems. In *International Workshop on Optimisation in Multi-Agent Systems (OptMAS@ AAMAS 2019)*, 2019.

[131] R. E. Schapire and Y. Freund. Boosting: Foundations and algorithms. *Adaptive Computation and Machine Learning*, 2012.

[132] Alvin Sebastian, Maolin Tang, Yanming Feng, and Mark Looi. A multicast routing scheme for efficient safety message dissemination in vanet. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6. IEEE, 2010.

[133] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. History and definitions. In *Self-organising software*, pages 33–74. Springer, 2011.

[134] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. *Self-organising Software*. Springer, 2011.

[135] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. Self-organising systems. In *Self-organising Software*, pages 7–32. Springer, 2011.

[136] Glenn Shafer. Dempster-shafer theory. *Encyclopedia of artificial intelligence*, 1:330–331, 1992.

[137] Zhenyu Shan, Yingjie Xia, Peipei Hou, and Jifeng He. Fusing incomplete multisensor heterogeneous data to estimate urban traffic. *IEEE MultiMedia*, 23(3):56–63, 2016.

[138] A. Sharif, Jian Ping Li, and Muhammad Asim Saleem. Internet of things enabled vehicular and ad hoc networks for smart city traffic monitoring and controlling: A review. *International Journal of Advanced Networking and Applications*, 10(3):3833–3842, 2018.

[139] Samir Shariff, M. Saad Alam, Zaurez Ahmad, Hafiz Malik, and Furkan Ahmad. Design and optimization of intelligent transportation system for smart cities. *SAE Technical Paper*, 2019(2019-01-0489), 2019.

[140] Gurinder Singh, Vishal Jain, Jyotir Moy Chatterjee, and Loveleen Gaur. *Cloud and IoT-Based Vehicular Ad Hoc Networks*. John Wiley & Sons, 2021. Google-Books-ID: tfMlEAAAQBAJ.

[141] J Sobieraj. *Methodes et outils pour la conception de Systèmes de Transport Intelligents Cooperatifs*. PhD thesis, Universite Paris-Saclay, 2018.

[142] J. Sobieraj, Guillaume Hutzler, and Hanna Klaudel. Modelisation et simulation de la cooperation dans les systèmes de transport intelligents : un comparatif. In *Journees Francophones sur les Systèmes Multi-Agents (JFSMA 2017)*, 2017.

[143] Jack Stilgoe. How can we know a self-driving car is safe? *Ethics and Information Technology*, pages 1–13, 2021.

[144] Geng Sun, Jiahui Li, Yanheng Liu, Shuang Liang, and Hui Kang. Time and energy minimization communications based on collaborative beamforming for uav networks: A multi-objective optimization method. *IEEE Journal on Selected Areas in Communications*, 2021.

[145] Lei Sun, Kailun Yang, Xinxin Hu, Weijian Hu, and Kaiwei Wang. Real-time fusion network for rgb-d semantic segmentation incorporating unexpected obstacle detection for road-driving images. *IEEE Robotics and Automation Letters*, 5(4):5558–5565, 2020.

[146] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. *Advances in Neural Information Processing Systems*, 26, 2013.

[147] Atena M Tabakhi, Yuanming Xiao, William Yeoh, and Roie Zivan. Branch-and-bound heuristics for incomplete dcops. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1677–1679, 2021.

[148] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. Pc-syncbb: A privacy preserving collusion secure dcop algorithm. *Artificial Intelligence*, 297:103501, 2021.

[149] Ozan Tonguz, Nawapom Wisitpongphan, Fan Bai, Priyantha Mudalige, and Varsha Sadekar. Broadcasting in vanet. In *2007 mobile networking for vehicular environments*, pages 7–12. IEEE, 2007.

[150] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[151] M. Di Vaio, G. Fiengo, A. Petrillo, A. Salvi, S. Santini, and M. Tufo. Cooperative shock waves mitigation in mixed traffic flow environment. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[152] Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

[153] Nicolas Verstaevel. *Self-organization of robotic devices through demonstrations*. Phd thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2016.

[154] Manuel Augusto Vieira, Manuela Vieira, Paula Louro, and Pedro Vieira. Vehicular communication in a crossroad using visible light. *Sensors & Transducers*, 245(6):41–48, 2020.

[155] Van Tan Vu, Olivier Sename, Luc Dugard, and Péter Gáspár. Enhancing roll stability of heavy vehicle by lqr active anti-roll bar control using electronic servo-valve hydraulic actuators. *Vehicle system dynamics*, 55(9):1405–1429, 2017.

[156] Maxime Wack, Tenda Okimoto, Maxime Clement, and Katsumi Inoue. Local search based approximate algorithm for multi-objective dcops. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 390–406. Springer, 2014.

[157] Yong Wang, Jian Yu, Shengxiang Yang, Shouyong Jiang, and Shuang Zhao. Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons. *Swarm and Evolutionary Computation*, 50:100559, 2019.

[158] Ziran Wang, Guoyuan Wu, and Matthew Barth. Distributed consensus-based cooperative highway on-ramp merging using v2x communications. In *WCX World Congress Experience*. SAE International, apr 2018.

[159] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 2000.

[160] Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.

[161] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2 edition, 2009.

[162] Feng Wu and Nicholas Jennings. Regret-based multi-agent coordination with uncertain task rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[163] Fangchun Yang, Shangguang Wang, Jinglin Li, Zhihan Liu, and Qibo Sun. An overview of internet of vehicles. *China communications*, 11(10):1–15, 2014.

[164] William Yeoh. *Speeding up distributed constraint optimization search algorithms*. Citeseer, 2010.

[165] Mahdi Zargayouna, Flavien Balbo, and Khadim Ndiaye. Generic model for resource allocation in transportation. application to urban parking management. *Transportation Research Part C: Emerging Technologies*, 71:538–554, 2016.

[166] Yanru Zhang and Ali Haghani. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58:308–324, 2015.

[167] Jens R Ziehn, Masoud Roschani, Miriam Ruf, Dennis Bruestle, Jürgen Beyerer, and Melanie Helmer. Imaging vehicle-to-vehicle communication using visible light. *Advanced Optical Technologies*, 9(6):339–348, 2020.