UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DENISE DE OLIVEIRA

Towards Joint Learning in Multiagent Systems Through Opportunistic Coordination

Ph.D. Thesis Proposal

Prof. Dr. Ana Lúcia C. Bazzan
Advisor

Porto Alegre, November 2007

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

GT	game theory
MASL	multiagent learning

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

Several approaches search for faster or more precise learning algorithms for independent agents. Other approaches search for agents coordination mechanisms, so the agents can act more efficiently on performing joint tasks. The learning mechanism for the agents in a group are usually not emphasized. Although learning and coordination in complex systems are closely related, the majority of the learning approaches is limited to systems where there is no need for a coordination of action or the exchange of information about the environment for performing joint tasks. This work presents some methods of multiagent reinforcement, group formation, and two complex domains where learning has not being completely explored. By those reasons, a new method of multiagent reinforcement learning is presented. In the proposed method, the agents try to act and learn in the most independent way as possible in the environment before they search for information or actions of other agents. The objective of the proposed approach is to have a representation of the states where the agent must have more information about the environment or to perform joint actions.

Keywords: Multiagent systems, reinforcement learning, coordination.

Em busca de Aprendizado Conjunto em Sistemas Multiagente Atravï¿½s de Coordenaï¿½ï¿½o Oportunï¿½stica

RESUMO

Diversas abordagens buscam algoritmos mais rï¿½pidos ou mais precisos de aprendizado para agentes independentes. Outras abordagens buscam mecanismos de coordenaï¿½ï¿½o entre os agentes para que estes possam agir de modo mais eficiente para a realizaï¿½ï¿½o de tarefas em conjunto, sendo que o aprendizado dentro do grupo normalmente nï¿½o ï¿½ enfatizado. Embora o aprendizado e coordenaï¿½ï¿½o em sistemas complexos esteja muito relacionada, a maioria das abordagens de aprendizado estï¿½ limitada a sistemas onde nï¿½o ï¿½ necessï¿½ria a coordenaï¿½ï¿½o de aï¿½ï¿½es ou a troca de informaï¿½ï¿½es sobre o ambiente para a realizaï¿½ï¿½o de tarefas em conjunto. Este trabalho apresenta alguns mï¿½todos de aprendizado por reforï¿½o multiagente, formaï¿½ï¿½o de grupos e dois domï¿½nios complexos onde o aprendizado ainda nï¿½o foi explorado completamente. Por essas razï¿½es, ï¿½ apresentado um novo mï¿½todo de aprendizado por reforï¿½o multiagente. No mï¿½todo proposto, os agentes buscam agir e aprender de modo mais independente possï¿½vel no ambiente antes de buscarem por informaï¿½ï¿½es ou aï¿½ï¿½es de outros agentes. O objetivo da abordagem proposta ï¿½ ter uma representaï¿½ï¿½o dos estados onde o agente deve ter mais mais informaï¿½ï¿½es sobre o ambiente ou para realizar tarefas conjuntas.

Palavras-chave: sistemas multiagente, aprendizado por reforï¿½o, coordenaï¿½ï¿½o.

1 INTRODUCTION

In several decision problems, agents must interact in order to perform tasks in complex and dynamic scenarios. Usually problems of this kind are characterized by an incomplete communication among the agents, by agents having a limited view, agents acting individually in several situations in the environment. Along with this text, will be shown that problems where the agent needs to learn receiving some information of the environment state and some kind of reward or penalty are hard to solve even for a single agent in a complex scenario. When a group of agents must interact, in a cooperative way or not, in a complex environment, a centralized and complete solution is usually not possible to achieve given the complexity.

Research in multiagent learning (MASL) is generally focused on relatively simple environments or on the game theoretic point of view. Game theory (GT) is a branch of mathematics that studies strategic interactions between agents in a game. A game is characterized by agents (players) making decisions trying to get the best possible reward (payoff). These games can be cooperative or non-cooperative. Cooperative games are the characterized by a group of agents, usually called *coallition*, is trying to get the highest common reward. A non-cooperative game is characterized by agents trying to receive the best possible individual reward, even if they cooperate in order to achieve their individual goals.

MASL in games searches for the best strategy when an agent has one or more opponents, usually searching for the equilibrium. An equilibrium point is a stable outcome of a game associated with the strategies where no changes in the strategies can improve the outcome. It is considered stable because if a player chooses unilaterally a different strategy he gets a lower payoff. Recently, a discussion on multiagent learning and its future as research was open with the paper entitled “If multi-agent learning is the answer, what is the question?” (SHOHAM; POWERS; GRENAGER, 2007). This article was the initial discussion point of the special issue of the Autonomous Agents and Multi-Agent Systems Journal on MASL, from August 2007 (number 1, volume 15). The discussion was basically focused on the game theoretic aspect of multiagent learning and made some important points. The authors pointed out that there is a need to be clear about the problem being addressed and the associated evaluation criteria. For this, they have identified five distinct agendas in multiagent learning:

Computational: views the learning algorithm as an iterative way to compute properties of the game;

Descriptive: the goal is to investigate formal models of learning that are similar to human behavior or with the behavior of other agents;

Normative: focuses on determining with sets of learning rules are in equilibrium with each other;

Prescriptive cooperative and non-cooperative the question here is how agents should learn in a repeated common payoff games (cooperative) or in a repeated independent payoff games (non-cooperative).

In (CLAUS; BOUTILIER, 1998), a study on the dynamics of reinforcement learning (RL) in cooperative multiagent systems (MAS) was made. Their work fits in the prescriptive cooperative agenda since they focus on the n-player repeated cooperative game structure and exploration strategies on convergence to equilibria.

Besides these agendas, the authors state that the field cannot advance while arbitrary learning strategies are created and analyzed considering whether the resulting dynamics converge in certain cases.

In response to the article from Shoham and colleagues, several articles have being written (SANDHOLM, 2007; GORDON, 2007; YOUNG, 2007; STONE, 2007). Sandholm proposed a refinement in the agendas of Shoham and colleagues, adding hierarchy and renaming the normative agenda to "learning algorithms in equilibrium". This proposed taxonomy can be seen in Figure 1.1. In addition to these agendas, Gordon proposed two additional agendas: "modeling" and "design". The objective of those agendas is to cover the problems that needs to be considered before the agents start the learning process.

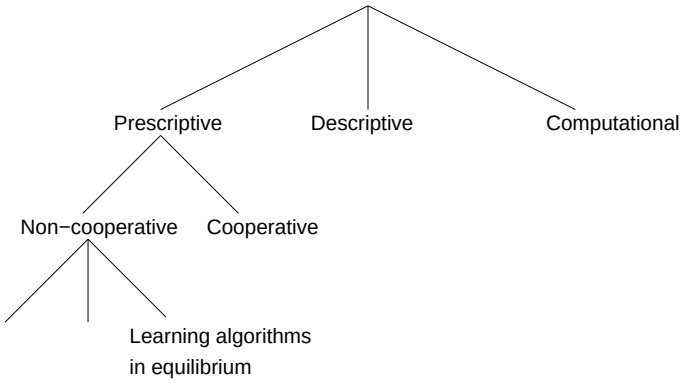


Figure 1.1: Hierarchy proposed in (SANDHOLM, 2007).

Peter Stone reinforces that multiagent learning is a good tie between game theory and artificial intelligence (AI) and there are several works in the intersection of these two areas. Multiagent interactions can be characterized as games, but in several cases, it is not only that the convergence to an equilibrium is not a goal, but that the very formulation of the encounter as a normal form or extensive form game, if even practical, does little to make progress towards a solution. From an AI perspective, multiagent learning should be considered more broadly than GT can address.

There are several multiagent learning papers involving a mere two agents and assuming fully cooperative environments, the field can be expanded to more realistic and interesting environments. In (PANAIT; LUKE, 2005), they show the directions of research that have not been adequately explored:

- Larger numbers of agents, in the range of 10-1000 or more;

- Team heterogeneity, most of the literature presumes that the agents are identical in behavior and in ability;
- Agents with internal state or other complexity;
- Dynamically changing teams and scenarios.

This work focuses on a not so explored area of research on multiagent reinforcement learning that is increasing the size of the observations space for independent and joint action learners (JALs). This research fits in the Shoham's descriptive agenda, since the objective is to find a better way for the agent to learn in complex environments. In one extreme there are independent learners (ILs) approaches that considers other agents as part of the environment. The concept of IL in this work is similar to the one presented in (CLAUS; BOUTILIER, 1998), where the agents ignore the existence of other agents, but here this definition not only applied for agents that use the Q-learning technique. In the opposite direction, there are joint action learners (JALs) that consider all observations and actions from every agent in the environment. These JALs have a huge state representation, since they model all states from all agents, even non significative states. The complexity of the states representation is a key factor the limits the use of the standard RL algorithms in complex problems, even for ILs.

State abstraction, or state aggregation, has been widely studied. Abstraction can considered as mapping of the original description of a problem to a more compact representation, allowing the decision maker to distinguish relevant information from irrelevant information. From a computational perspective, state abstraction is a technique for making possible to apply learning algorithms in large problems. (LI; WALSH; LITTMAN, 2006) presents a review of those methods, dividing the different approaches according whether the algorithm needs to have full knowledge of the MDP and the exactness of the method. They show that there is a tradeoff between maximizing state space reduction and minimizing information loss when selecting abstractions. Another issue is that abstract state space can be almost as large as the ground state space in complex domains. In summary, there are several techniques to deal with multiagent learning in several situations but, until now, there is no learning method that uses an adaptive modification of the agents' knowledge representation, adding new possible states and joint actions according to the its experiences.

This work proposes a new approach where the agent starts using a simple individual representation and expand it to a more complex environment representation, allowing the agent to increase its perception of the environment and have a better performance. The agent acts like a IL in states where only its own information is used, acts like a IL with shared sentations in states where it uses information received from partners (agents in a group) and tryes to act like a JAL. This work searches form a trade-off solution using useful characteristics from both extremes in order to have a learning mechanism able to cope with different aspects of complex environments.

This work proposes a multiagent learning approach using independent learning before increasing the states space or performing joint actions.

The proposed approach provides a multiagent learning framework that reduces the

state-space of joint actions, trying to achieve the best possible solution in domains with the following characteristics:

- Agents cooperate when it is profitable; otherwise, they look for the action with the highest profit. In other words, all agents are alway s trying to find the best possible reward.
- The problem must be solved in a distributed or partially distributed way, using communication among the agents.
- There must exist at least one task where the agent is not capable of achieving the best reward acting in a independent way.
- The environment must have the following characteristics:
  - **Dynamic**: the environment changes are not only consequences of the agents actions, changes are beyond the agents control;
  - **Non-deterministic**: each independent action may have more than one possible effect, since the effect is a result of its own action, other agents actions and external factors;
  - **Discrete**: each agent have a finite number of possible actions and observable states;
  - **Partially observable**: each agent has a limited perception of the current environment state.

The objective of the proposed approach is to have a representation of the states where the agents must have more information about the environment or to perform joint actions. The agent starts with no previous knowledge about other agents, not even if there exists any other agents in the scenario. Different from other approaches, that have assumptions such as: pre-defined joint actions or dependencies among agents are known beforehand; in this approach, it is assumed that the agents have no knowledge about their inter-dependencies. Given those restrictions, a group formation method is needed in order to have a communication and an action group. A group formation approach must be chosen according to their specific characteristics, since some methods are more suitable for some environments then others.

This work is organized as follows:

Chapter 2 presents review on multiagent learning and mathematical models of sequential decision making in stochastic domains. Centralized and decentralized formalizations are presented, including some complexity analysis of the models.

Chapter 3 presents three group formation methods: coalition, teams and clustering. Those methods must be studied and compared since the proposed MAS learning method needs a group of agents exchanging information about the environment. At the end of this chapter, there is a comparative analysis and a discussion about the application of those methods.

Chapter 4 two dynamic domains are presented: urban traffic control (UTC) and search and rescue (SAR). The first part has an introduction on traffic lights control and some previous approaches on this domain. The second part of this chapter presents the RoboCup Rescue simulator and its analysis. As the final part of this chapter, there is a comparison and a discussion about both domains.

Chapter 5 introduces the proposed approach and the initial algorithm. The method applies MAL using group formation to exchange information and perform actions. At the end of this chapter, the application of the proposed approach on the domains presented on Chapter 4, UTC and SAR, is discussed.

Chapter 6 presents some initial conclusions, an activities plan and a schedule for the next steps of this work.

2 MULTIAGENT REINFORCEMENT LEARNING

The standard reinforcement learning framework consists of an agent interacting repeatedly with the environment at discrete intervals (KAELBLING; LITTMAN; MOORE, 1996). At each time step, the agent perceives the environments current state and selects an action. The environment responds with a reward signal and a new state. According to Sutton and Barto (SUTTON; BARTO, 1998) the two most important features of reinforcement learning are the trial-and-error search and the delayed reward. In this Chapter, a brief review on Markov Decision Process (MDP) will be presented, since they are the formalism used in the reinforcement learning methods presented.

2.1 Review on Markov Decision Processes

A Markov Decision Process, according to (PUTERMAN, 2005), consists of five elements: decision epochs, states, actions, transition probabilities between states, and rewards. It models an agent acting in a stochastic environment where the objective is to maximize its long term reward, (KAELBLING; LITTMAN; CASSANDRA, 1998), and serve as a framework for solving several problems. The agent receives a world state and responds with some action, those actions affect the environment state.

We can extend the MDP definition to problems where the environment is not completely observable, Partially Observable Markov Decision Processes (POMDP). Those models can be defined as a tuple $\langle S, A, P, R, \Omega, O, T \rangle$, where:

S is a finite and discrete set of states;

A is a finite and discrete the set of actions;

P ($P(s'|a,s)$) is the probability of the environment transitioning to state s', given it was in state s and took action a;

R is a reward function, that is received for being in state s and taking action a ($R(s,a)$);

Ω is a finite set of observations;

O is the observation probabilities. $O(s,a,s',o)$ the probability of agent seeing observation o, taking action a, given the current state is s and the next state is s';

T is the horizon, whether infinite or infinite, a positive integer. When T is infinite a discount factor, $0 \leq \gamma < 1$, is used.

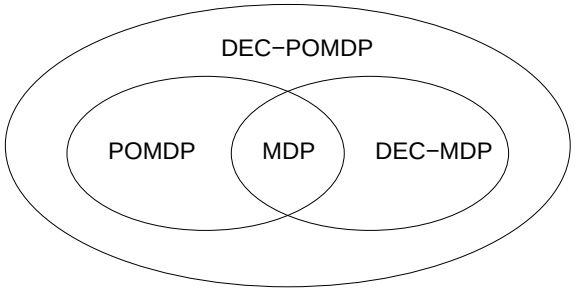


Figure 2.1: Relation among the models

Similar to the POMDP definition, a DEC-POMDP can be described as a tuple $M = \langle S, A_1 \ldots A_n, P, R, \Omega_1 \ldots \Omega_n, O, T \rangle$, where:

S a finite and discrete set of states;

$A_1 \ldots A_n$ is a set of agents actions for agent, from $1 \ldots n$;

R global reward function;

$\Omega_1 \ldots \Omega_n$ the set of observations for agent, from $1 \ldots n$.

O the observation probabilities. $O(s, a, s', o)$ the probability of agent seeing observation o, taking action a, given the current state is s and the next state is s';

T the horizon, whether infinite or infinite, a positive integer. When T is infinite a discount factor, $0 \leq \gamma < 1$, is used.

The multiagent extension of the single-agent POMDP model with collective partial observability is a decentralized POMDP (DEC-POMDP). Figure 2.1 shows the relation among the models. This model assumes that the observations of the agents are uncertain and given by a probability which depends on the previous state, current state, and previous performed joint action. It assumes each agent receives a shared reward. A joint policy is defined as a tuple of local policies: $\delta = \langle \delta^1, \ldots, \delta^m \rangle$. The objective is to find a joint policy that maximizes the expected total reward. The model that assumes that each agent has a different reward function is called Partially Observable Stochastic Game (POSG), presented in (HANSEN; BERNSTEIN; ZILBERSTEIN, 2004).

The complexity of solving a DEC-POMDP falls in the complexity class NEXP-complete (BERNSTEIN et al., 2002), and several simplified models exist that make additional assumptions about the parameters of the model. A reduction in complexity is seen when the agents are mostly independent (BECKER et al., 2003).

A collaborative DEC-MDP can be regarded as one large single agent in which each joint action is represented as a single action. It is then possible to apply a model-free reinforcement-learning technique like Q-learning, that will be described in the next Section. This approach eventually results in an optimal joint policy. In this MDP learners approach either a central controller models the complete MDP and communicates to each agent its individual action, or each agent models the complete MDP separately and selects the individual action that corresponds to its own identity. In the latter case, the agents do not need to communicate but they have to be able to observe the executed joint action and the received individual rewards. However, the distributed nature of the problem requires the agents to explore at the same time.

Although this approach leads to the optimal solution, it is infeasible for problems with many agents for several reasons. The first reason is that is intractable to model the complete joint action space, which is exponential in the number of agents. For example, a problem with n agents, each able to perform m actions, results in m^n Q-values per state. Secondly, the agents might not have access to the needed information for the update because they are not able to observe the state, actions and reward of all other agents.

2.2 Single Agent Reinforcement Learning

This section presents some methods were the agents do not model other agents states or actions directly, they consider other agents as part of the environment. Reinforcement learning methods can be divided into two categories: model-free and model-based. Model-based methods assume that the transition function T and the reward function R are available. Model-free systems, such as Q-learning (WATKINS; DAYAN, 1992), on the other hand, do not require that the agent have access to the environment's model.

2.2.1 Q-learning

Q-Learning works by estimating good state-action values, the Q-values, which are a numerical estimator of quality for a given pair of state and action. More precisely, a Q-value $Q(s,a)$ represents the maximum discounted sum of future rewards an agent can expect to receive if it starts in s, chooses action a and then continues to follow an optimal policy.

Algorithm 2.1: Q-learning

```
1 Initialize Q(s,a) arbitrarily ;
2 foreach episode do
3     Initialize s;
4     repeat for each step of episode
5         Choose a from s using a policy derived from Q;
6         Take action a, observe s' and r;
7         Q(s,a) ← Q(s,a) + α (r + γ max_a' Q(s',a') − Q(s,a)) ;
8         s ← s' ;
9     until s is terminal;
```

Q-Learning algorithm approximates the Q-values $Q(s,a)$ as the agent acts in a given environment. Uses an update rule, at line 7 in the Algorithm 2.1, for each experience tuple $\langle s,a,s',r\rangle$, where α is the learning rate and γ is the discount for future rewards. As can be seen in the update rule, the estimation of the Q-values does not rely on T or R, as Q-learning is model free. When the Q-values are nearly converged to their optimal values, it is appropriate for the agent to act greedily, that is, to always choose the action with the highest Q-value for the current state.

2.2.2 Prioritized Sweeping

Prioritized Sweeping (PS) is somewhat similar to Q-Learning, except for the fact that it continuously estimates a single model of the environment. Also, it updates more than one state value per iteration and makes direct use of state values, instead of Q-values. The states whose values should be updated after each iteration are determined by a priority

queue, which stores the states *priorities*, initially set to zero. Also, each state remembers its predecessors, defined as all states with a non-zero transition probability to it under some action. At each iteration, new estimates $\hat{T}$ and $\hat{R}$ of the dynamics are made. The usual manner to update $\hat{T}$ is to calculate the maximum likelihood probability. Instead of storing the transitions probabilities in the form $T(s,a,s')$, PS stores the number of times the transition (s,a,s') occurred and the total number of times the situation (s,a) has been reached. This way, it is easy to update these parameters and to compute $\hat{T}$ as $\frac{|(s,a,s')|}{|(s,a)|}$.

Given an experience tuple $\langle s,a,s',r\rangle$, PS behaves according to Algorithm 2.2.

Algorithm 2.2: Prioritized Sweeping

1 $V_{old}(s) \leftarrow V(s)$;
2 Update the state's value ;
3 $V(s) \leftarrow max_a \left(\hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s')V(s')\right)$;
4 $priority(s) \leftarrow 0$;
5 Compute the value change: $delta \leftarrow |V_{old} - V(s)|$;
6 Use δ to modify the priorities of each predecessors s_p of s:
$priority(s_p) \leftarrow \delta \hat{T}(s_p,a_p,s)$ where a_p is any action such that $\hat{T}(s_p,a_p,s) \geq 0$

2.2.3 RL-CD

On non-stationary environments, both the model-free and the model-based RL approaches need to re-learn at each change in the environment dynamics. The policy calculated for a given environment is no longer valid after a change in dynamics. This causes a performance drop during the readjustment phase, even for previously experienced environment dynamics.

In order to cope with non-stationary environments, alternative RL methods have been proposed, e.g. the mechanisms proposed by Choi and colleagues (CHOI; YEUNG; ZHANG, 2001) and Doya and colleagues (DOYA et al., 2002). Unfortunately, their approaches require a fixed number of models, assuming that the approximate number of different environment dynamics is given beforehand. Since this assumption is not always realistic, the RL-CD (Reinforcement Learning with Context Detection) method, (SILVA et al., 2006), tries to overcome it by incrementally building new models.

RL-CD, assumes that the use of multiple partial models of the environment is a good approach for dealing with non-stationary scenarios. The use of multiple models makes the learning system capable of partitioning the knowledge in a way that each model automatically represents one model. To each model, it is assigned an optimal policy and a trace of prediction error of transitions and rewards, responsible for estimating the quality of a given partial model. Moreover, the creation of new models is controlled by a continuous evaluation of the prediction errors generated by each partial model. In the following subsections describes how the contexts are learnt, and then is shown how to detect and switch to the most adequate model given a sequence of observations.

A context consists of a given environment dynamics, which is experienced by the agent as a class of transitions and rewards. The mechanism for detecting context changes relies on a set of partial models for predicting the environment dynamics. A partial model m contains a function T_m, which estimates transition probabilities, and also a function R_m, which estimates the rewards to be received.

For each partial model, classic model-based reinforcement learning methods such as

Prioritized Sweeping may be used to compute a locally optimal policy. The policy of a partial model is described by the function $\pi_m(s)$ and it is said to be locally optimal because it describes the optimal actions for each state in a specific context. For example, if the dynamics of a non-stationary environment Θ can be described by m_1, then π_{m_1} will be the associated optimal policy. If the non-stationarity of Θ makes itself noticeable by making π_{m_1} suboptimal, then the system creates a new model, m_2, which would predict with a higher degree of confidence the transitions of the newly arrived context. Associated with m_2, a locally optimal policy π_{m_2} would be used to estimate the best actions in m_2. Whenever possible, the system reuses existing models instead of creating new ones.

Given an experience tuple $\langle s, a, s', r \rangle$, the current partial model m is updated by adjusting its model of transition and rewards by T_m, using Equation 2.1. The reward model R_m is updated using Equation 2.3.

$$T_m(s,a,\kappa) = T_m(s,a,\kappa) + \Delta T_m(\kappa), \quad \forall \kappa \in S \tag{2.1}$$

$$\Delta T_m(\kappa) = \begin{cases} \dfrac{1 - T_m(s,a,\kappa)}{N_m(s,a)+1} & \kappa = s' \\ \dfrac{0 - T_m(s,a,\kappa)}{N_m(s,a)+1} & \kappa \neq s' \end{cases} \quad \forall \kappa \in S \tag{2.2}$$

$$R_m(s,a) = R_m(s,a) + \Delta R_m \tag{2.3}$$

$$\Delta R_m = \frac{r - R_m(s,a)}{N_m(s,a)+1} \tag{2.4}$$

The quantity $\mathcal{N}_m(s,a)$, computed by Equation 2.5, reflects the number of times, in model m, action a was executed in state s. $\mathcal{N}_m$ is computed considering only a truncated (finite) memory of past M experiences.

$$\mathcal{N}_m(s,a) = min\Big(\mathcal{N}_m(s,a) + 1, M\Big) \tag{2.5}$$

A truncated value of $\mathcal{N}$ acts like an adjustment coefficient for T_m and R_m, causing transitions to be updated faster in the initial observations and slower as the agent experiments more. Having the values for T_m and R_m, the transition probabilities are updated according to Equation 2.1, where $\Delta T_m(\kappa)$ is defined by equation 2.2. The model of expected rewards is also updated using Equation 2.3, where ΔR_m is defined by Equation 2.4.

In order to detect changes in the environment, the system must be able to evaluate how well the current partial model can predict the environment. Thus, an error signal is computed for each partial model. The *instantaneous error* is proportional to a *confidence value*, which reflects the number of times the agent tried an action in a state. Given a model m and an experience tuple $\langle s, a, s', r \rangle$, the instantaneous error e_m and the confidence $c_m(s,a)$ are calculated using Equations 2.6 and 2.7, respectively.

$$c_m(s,a) = \frac{N_m(s,a)}{M} \tag{2.6}$$

$$e_m = c_m(s,a)\Big(\Omega \;\; e_m^R + (1-\Omega) \;\; e_m^T\Big) \tag{2.7}$$

Where Ω specifies the relative importance of the reward and transition prediction errors for the assessment of the model's quality. The instantaneous quality of reward prediction is given by Equation 2.8 and the instantaneous quality of the transition prediction is given by Equation 2.9.

$$e_m^R = 1 - 2\ \ (Z_R(\Delta R_m)^2) \tag{2.8}$$

$$e_m^T = 1 - 2\ \ (Z_T \sum_{\kappa \in \mathscr{S}} \Delta T_m(\kappa)^2) \tag{2.9}$$

Where Z_R and Z_T are normalized factors given by:

$$Z_R = (\frac{N(s,a)+1}{R_{max} - R_{min}})^2$$

and

$$Z_T = \frac{1}{2}(N(s,a)+1)^2$$

Once the instantaneous error has been computed, the trace of prediction error E_m for each partial model is updated using Equation 2.10, where ρ is the adjustment coefficient for the error.

$$E_m = E_m + \rho\left(e_m - E_m\right) \tag{2.10}$$

Algorithm 2.3: RL-CD

m_{cur} is the currently active partial model ;
$\mathscr{M}$ the set of all available models ;
s is the current state ;
$m_{cur} \leftarrow newmodel()$;
$\mathscr{M} \leftarrow \{m_{cur}\}$;
$s \leftarrow s_0$, where s_0 is any starting state ;
repeat for each *step of episode*
    Let a be the action chosen by $\pi_{m_{cur}}(s)$;
    Observe next state s' and reward r;
    Update E_m, for all m, according to Equation 2.10;
    $m_{cur} \leftarrow \arg\min_m (E_m)$;
    if $E_{m_{cur}} < E_{min}$ **then**
        $m_{cur} \leftarrow newmodel()$;
        $\mathscr{M} \leftarrow \mathscr{M} \cup \{m_{cur}\}$;
    Update $T_{m_{cur}}$ and $R_{m_{cur}}$(Equations 2.1 and 2.3) ;
    $N_m(s,a) \leftarrow \min(N_m(s,a)+1, M)$;
    Update policy or state values, according to the specific RL algorithm that is being used ;
    $s \leftarrow s'$;
until *s is terminal*;

The error E_m is updated after each iteration for every partial model m, but only the active model is corrected according to Equations 2.1 and 2.3. A plasticity threshold λ is used to specify until when a partial model should be adjusted. When E_m becomes higher than λ, the predictions made by the model are considered sufficiently different from the real observations. In this case, a context change is detected and the model with lowest error is activated. A new partial model is created when there are no models with trace error smaller than the plasticity. The mechanism starts with only one model and then incrementally creates new partial models as they become necessary. Pseudo-code for RL-CD is presented in Algorithm 2.3.

The *newmodel*() routine is used to create a new partial model and initializes all estimates and variables to zero, except T_m, initialized with equally probable transitions. The values of parameters M, ρ, Ω and λ must be tuned according to the problem. Small values of ρ are appropriate for noisy environments; higher values of M define systems which require more experiences in order to gain confidence regarding its predictions; in general applications, Ω might be set to 0.5; the plasticity λ should be set to higher values according to the need for learning relevant (non-noisy) but rare transitions.

2.3 Independent vs. Cooperative Agents

In (TAN, 1993), three ways of agent cooperation are identified: information, episodic experience, and learned knowledge. The main thesis of Ming Tan is:

> *If cooperation is done intelligently, each agent can benefit from other agents' instantaneous information, episodic experience, and learned knowledge.*

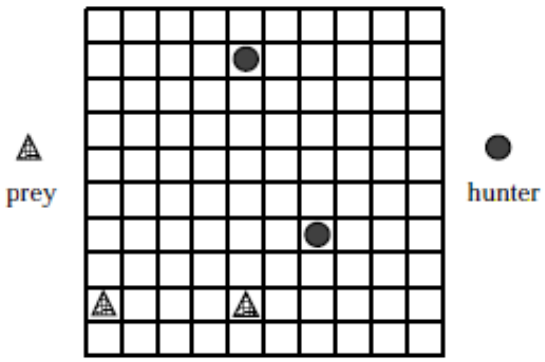


Figure 2.2: A 10x10 grid worl (TAN, 1993).

The investigation is done in a hunter-prey scenario, where agents are seeking to capture a randomly-moving prey in a 10 by 10 grid world, as shown in 2.2. On each time step, each agent has four possible actions: moving up, down, left, or right within the boundary. More than one agent can occupy the same cell. A prey is captured when it occupies the same cell as a hunter or two hunter either occupy the same cell as the prey or are next to the prey. Upon capturing a prey, the hunters involved receive $+1$ reward. Hunters receive -0.1 reward for each move when they do not capture a prey. Each hunter or scout has a limited visual field.

2.3.1 Case 1: Sharing Sensation

First the effect of sensation from another agent is studied. To isolate sensing from learning this is done on a one-prey/one-hunter task with a scouting agent that cannot capture prey. The scout makes random moves. At each step, the scout send its action and sensation back to the hunter. The sensation inputs from the scout are used only if the hunter cannot sense any prey.

Table 2.1: Scouting vs No Scouting

Hunter Visual Depth	Scout Visual Depth	Avg. Steps to capture a Prey	
		Training	Test
2	-	47.14	49.49
2	2	46.33	42.91
2	3	39.78	32.08
2	4	32.67	25.07

Table 2.3.1 gives the average number of steps to capture a prey in training, where the hunters' visual-field depth was 2 and varying the scout visual depth. As the scout's visual-field depth increases, the difference in their performances becomes larger. After verifying that the scout information helps the hunter, this concept was extended to hunters that perform both scouting and hunting. The results are summarized on Table 2.3.1.

Table 2.2: Independent vs. Mutual-scouting Agents.

Agent Type	Visual Depth	Avg. Steps to Capture a Prey	
		Training	Test
Independent	2	20.38	24.04
Mutual-scouting	2	25.20	24.52
Independent	3	14.65	16.04
Mutual-scouting	3	14.02	12.98
Independent	4	12.21	11.53
Mutual-scouting	4	11.05	8.83

2.3.2 Case 2: Sharing Policies or Episodes

In this case, is assumed that the agents do not share sensations. The question in this case is: If the agent can complete a task alone, cooperation is still useful? For answering this question, exchanging policies and exchanging episodes are studied. An episode is a sequence of sensation, action, and reward experienced by an agent. ,from (TAN, 1993) An episode is exchanged between the agent that has accomplished the task and its partner, so the experience would be duplicated. The second possibility is to learn from an expert agent. In the experiments, summarized in Figure 2.3 the expert change seems to be more effective.

2.3.3 Case 3: Joint Tasks

In this case, the hunter can capture a prey only with other agent. The cooperation can occur by passive observation or by active exchange of perceptions and location. The simulation results indicates that agents of the independent learning behavior tend to ignore the other and approach directly to the prey. In the passive observation case (where the

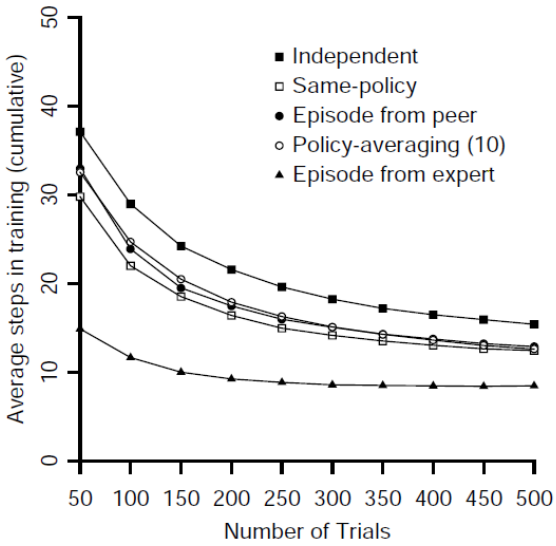


Figure 2.3: Results Summary.

agents perception is extended to include the partner location) the agents have a more effective learning,Figure 2.4. The mutual cooperation increases the states space without an increase in the state representation, having a slower but more effective learning process.

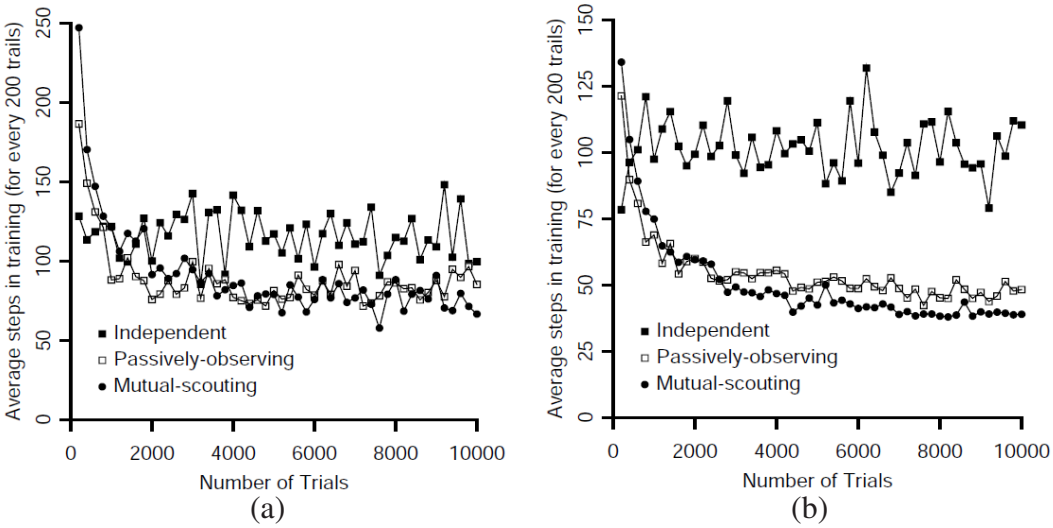


Figure 2.4: Runs for 2-prey/2-hunter (a) and 1-prey/2-hunter (b) joint task.

2.3.4 Conclusions

Cooperative reinforcement learning agents can learn faster and converge sooner than independent agents by sharing learned policies or solution episodes. On the other hand, the experiments shows that extra information can have a negative interference in the agent learning if the information is unnecessary. These trade-offs must be considered for autonomous and cooperative learning agents.

Some important issues of multiagent reinforcement learning are also presented, such as:

- Sensation must be selective because the size of a state space.

- One needs to use generalization techniques to reduce a state space and improve performance for complex, noisy tasks

- Information exchanging among agents incurs in communication costs.

2.4 Sparse and Context-specific Q-learning

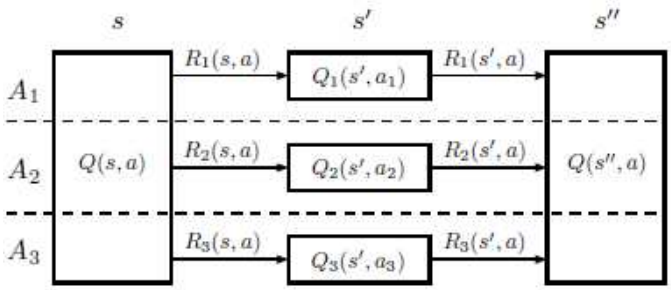


Figure 2.5: Q-tables where state s' is an uncoordinated state.

Sparse multiagent Q-learning is a reinforcement-learning technique which models context-specific coordination requirements (KOK; VLASSIS, 2004, 2006; KOK, 2006). The basic idea is to label each state of the system either as a coordinated or an uncoordinated state. In the coordinated states the agents learn based on joint actions, while in the uncoordinated states they learn independently.

In many situations only some of the agent shave to coordinate their actions. Now, we describe context-specific sparse cooperative Q-learning a technique which enables subsets of agents to learn how to coordinate based on a predefined coordination structure that can differ between states. This context-specific Q-learning uses a coordination graph approach (GUESTRIN; LAGOUDAKIS; PARR, 2002). In this approach, the Q-values are represented by value rules that specify the agents' coordination dependencies in particular states.

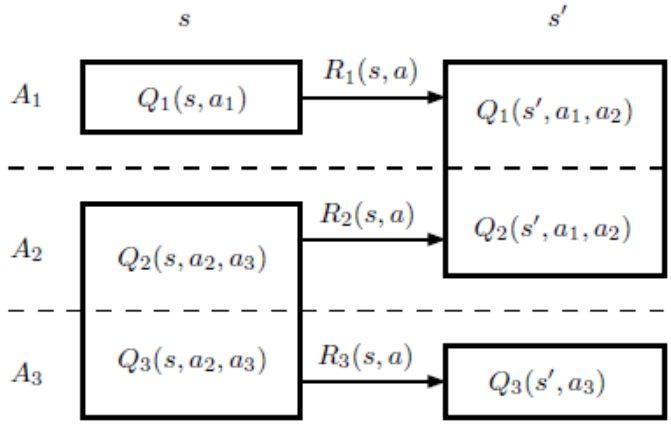


Figure 2.6: Representation of the Q components of three agents for a transition from state s to s', from (KOK; VLASSIS, 2004).

Table 2.3: Comparison of the Methods

Method	Avg. Time	#Q-values
Independent	16.86	97020
Manual policy	14.43	-
Sparse Coop.	13.14	32190
MDP Learners	12.87	242550

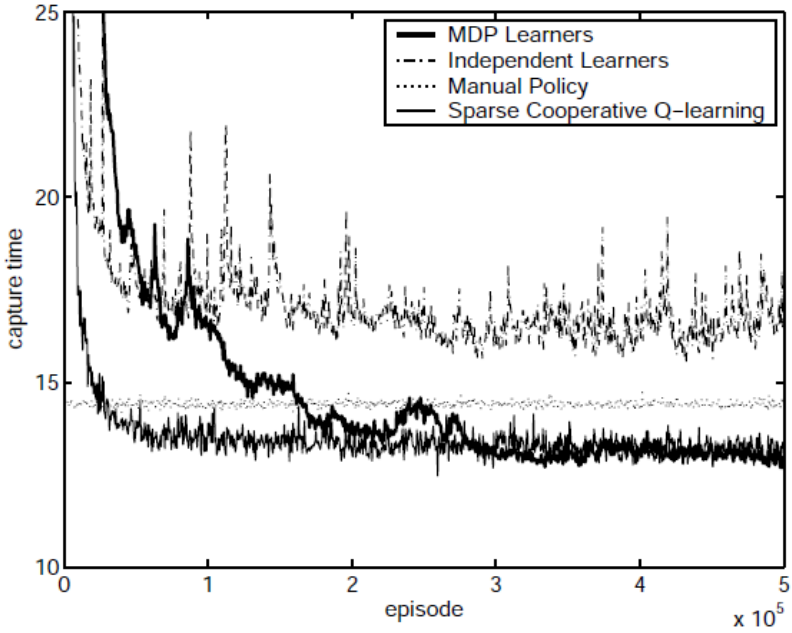


Figure 2.7: Capture time for the 4 different methods during the first 500,000 episodes (KOK; VLASSIS, 2004).

2.4.1 Conclusions

The approach was tested in the same 2 hunter 1 prey scenario seen on Section 2.3 In Table 2.4 we can see the average capture times for four different approaches for the last 10 test runs in a simulation and a manual implementation in which both predators first minimize the distance to the prey and then wait till both predators are located next to the prey. When both predators are located next to the prey, one of the two predators moves to the prey position. The context-specific learning approach converges to a slightly higher capture time than that of the MDP Learners. The explanation given by the authors for this difference is the fact that not all necessary coordination requirements are added as value rules.

2.5 Summary

This chapter was a brief review on multiagent system learning. The objective was to present some of the current approaches and to point out issues on those approaches. On analyzing those models, it is visible the need for more dynamic and adaptive forms of learning in complex systems. Usually, the considered scenarios are very restricted and do not represent real-life problems.

3 GROUP FORMATION

In this chapter we study some mechanisms for coalition formation, team formation, group formation using clustering and a discussion about those approaches. All those mechanisms try to have a group of agents that must act together for performing a task. The group formation can last until a single task is finished (one-shot), be permanent (created once) or dynamically change. The objective of this chapter is to show the characteristics of those different group formation mechanisms and to analyze how those mechanism can be used together with learning mechanisms.

3.1 Coalitions

A coalition is a group of agents cooperating in a joint action or to achieve a goal. Usually, the problem of coalition formation is not limited to cooperation for performing a single task, but multiple tasks and multiple agents, where agents try to cooperate within a competitive environment. This introduces a major complexity problem. Given n agents and m tasks, the number of possible coalitions is $m(2^n - 1)$, and the number of coalition configurations is $O(n^{n/2})$. This alliance can be for just one action (one-shot) or maintained for a longer period.

According to (SHEHORY, 2004), some questions should be considered when designing a coalition formation mechanism:

- Given a set of tasks and the other agents, which coalitions should an agent attempt to form?
- Once the preferred coalitions are known, what mechanisms can the agent use to form these coalitions?
- What guarantees does the selected coalition formation mechanism provide with regards to complexity, stability, immunity to attacks, fairness?
- Once a coalition has formed, how should its members distribute the work, and the gains, among themselves?
- How should the coalition be maintained and dissolve?

The answers for these questions depends on the type of the agents and the environment for which the coalition formation mechanism will be applied. In those aspects there are some basic characteristics that should be considered

Benevolence vs. self-interest Benevolent agents search for a global payoff maximization and self-interested players (agents) try to maximize their payoffs. Usually, coalitions mechanisms with benevolent agents are simpler then with mechanisms for self-interested ones.

Super-additivity and Monotonicity Super-additive environments are those in which, for any pair of coalitions, the value of the unified coalition is greater or equal to the sum of the values of separately coalitions. Monotonicity is when larger coalitions gain more then smaller ones. Given two coalitions A and B and $v(A)$, $v(B)$ are the values of coalition A and B respectively, super-additivity is when:

$$v(A \cup B) \geq v(A) + v(B)$$

And monotonicity is when:

$$A \subseteq B \Rightarrow v(A) \leq v(B)$$

Centralization vs. distribution Decentralization introduces communication overheads and may increase the complexity.

Complex domains, as the ones that will be presented on Chapter 4, cannot assume supper-additivity, neither assume benevolent agents or complete centralization. Given these restrictions, we present some coalition mechanisms on this section.

3.1.1 Game-theoretic approach

Game theory provides several solutions for coalitions of self-interested agents in non-super-additive environments. These solutions are not concerned with the coalition formation process but in analyzing the possible coalitions and checking its stability or fairness using concepts such as Kernel and Core.

3.1.2 Learning for coalition formation

There exists two models for dynamic coalition learning, the first uses case-based reinforcement learning (CBRRL), (SOH; LI, 2004), and the second uses Bayesian reinforcement learning, (CHALKIADAKIS; BOUTILIER, 2007, 2004). In this section we briefly introduce both models.

3.1.2.1 Coalition formation model using case-based reinforcement learning

Propose a multi-phase coalition formation model (MPCF) in which case-based reinforcement learning (CBRL) is applied to each agent have the ability to form coalitions in dynamic, uncertain, real-time, and noisy environments. The model consists of three phases:

1. Generates a plan to form the intended coalition;

2. Carries out the plan through candidate selection and multiple concurrent negotiations;

3. Evaluates the coalition formation process and quality of the coalition to learn to improve on future formation activity.

At the first phase, the coalition initiating agent generates a plan using case-based reasoning (CBR). A case consists of a problem description, a solution, and an outcome. The problem description refers to the task to be accomplished. The solution refers to the coalition formation plan, and a task description consists of: priority, set of sub-tasks, temporal relationships of its sub-tasks, time requirement, resource requirement, success rates, solution qualities, and a set of acceptable outcomes with corresponding utility values. A plan consists of: number of coalition candidates, number of coalition members, time allocated for coalition instantiations, allocation algorithm, number of messages recommended, expected coalition instantiation outcome, and the expected coalition execution outcome.

The learning mechanism is case-based learning (CBL) and tightly coupled with the coalition evaluation stage. CBR is used to retrieve the best case, and adapt its solution for the current coalition problem. The retrieval is based on the similarity in the problem description and the value of the case. This is how the element of reinforcement learning in put into CBR and subsequently CBL. The value of a case is based on the Q-value of applying the plan to the problem description.

In the second phase the agent allocates tasks or resources, identifying and ranking potential candidates, recruiting the candidates and coordinating agreements. If a coalition is formed, the agent executes it, this execution can be successful or a failure. Using this result, the agent evaluates its coalition formation process at the first and second stage trying to improve it. The evaluation is based on two groups of parameters: one for the efficiency of the planning and instantiation process, and one for the result of the executed coalition.

This learning framework involves multiple levels embedded in the three stages. At a low level on strategic and tactical details and at the meta-level, the agent learns to improve its planning and execution of the plan. The model uses an approach that synthesizes reinforcement and case-based learning.

3.1.2.2 Bayesian reinforcement learning for Coalition Formation

Propose a coalition formation model in which agents must derive collisional values by reasoning about the types of other agents and the uncertainty in the actions a coalition may take. The Bayesian core (BC) is presented as a new stability concept, and describe a dynamic coalition formation process that will converge to the BC if it exists.

The model uses a specific Bayesian RL model (CHALKIADAKIS; BOUTILIER, 2003) in which agents refine their beliefs about others through repeated interaction. In this coalition formation model, the agents maintain explicit beliefs about the types of others, and choose actions and coalitions considering their immediate value and also their value of information.

3.1.3 Maximal Clique Based Distributed Coalition Formation

In (TOSIC; AGHA, 2005) a graph algorithm, Maximal Clique Based Distributed Coalition Formation (MCBDCF), is proposed for coalition formation. An undirected graph represents the communication network topology among the agents. Each agent is a node in the graph. The edges exist if the two nodes are able to directly communicate to each other. The basic idea is to efficiently and in a fully decentralized manner partition this graph into (preferably, maximal) cliques of nodes.

The coalitions are maintained until they are no longer useful or meaningful. For instance, the coalitions should be transformed, or dissolved, when the interconnection topology (graph) considerably changes. Another possible reason to abandon the existing

coalition structure is when the agents determine that the coalitions have accomplished the set of tasks that these coalitions were formed to address.

The proposed distributed coalition formation algorithm is based on two main ideas:

1. Formulate a distributed task and/or resource allocation problem as a (distributed) set covering problem or a (distributed) set partitioning problem. Two (or more) coalitions overlap if there exists an element that belongs to both (all) of them.

2. Ensure that the formed coalitions of agents meet the robustness and fault tolerance criteria. The most robust coalitions of agents of a particular size are those that correspond to maximal cliques in the underlying interconnection topology of the agents' communication network.

The MCBDCF algorithm can be used as a subroutine scenarios where the system needs to reconfigure itself, and the agents need to form or transform coalitions in a fully distributed manner.

3.2 Teams

The majority of approaches for group formation focuses on creating short-term groups, that are united for executing a single task. In human societies, groups usually form long-term bindings. In (RATHOD; JARDINS, 2004), those two kind of group formation and also a mixed form of groups are used to analyze the performance of several strategies of group formation. Their work will be presented here.

Consider that the environment has a fixed number N of agents and it is simple enough that its characteristics do not affect the group formation process. The abilities are defined in a set from 1 to C different abilities, each agent can have exactly one ability. Each agent have only three possible states: unemployed, team leader and team member. Tasks are inserted at a regular rate in the system, each one of those tasks require a set of specific abilities and have a expiration time.

3.2.1 Communication

Agents need to communicate in order to be able to form teams, respecting following constraints:

1. A team leader can communicate with every agent in the system. This intent to be analogous to a real-world organization, in which the leader of the organization can typically communicate with its own employees as well as other leaders;

2. An unemployed agent can communicate with any team leader;

3. A team member who is not a leader can communicate with any team leader. This interaction can result in self-interested, non-stable behaviors where agents put their own interests ahead of their teams';

4. All agents can communicate with the controller agent.

3.2.2 Strategies

There are two different strategies: dynamic and stable team strategy. The first strategy is the base case, in which agents form short-term teams for the duration of the task.

Initially, all agents are unemployed. Task are introduced to the system, is then awarded to one of the bidders randomly. Once the task is awarded to an agent, it has to form a team by recruiting all the required agents and start working on the task before it expires. Once the task is completed, the task pay-off is equally distributed among the members. The team is dissolved and all the agents become unemployed again, ready to bid for tasks and join other teams. If the task expires before it is started, the team is not awarded any pay-off, the leader is made to pay a non-completion penalty, and the team is dissolved.

In the stable teams strategy, the agents try to form teams to accomplish the tasks with the objective of the best possible team reward. Five strategies of this kind are presented:

Naïve, Cautious the team is dissolved when its relative performance is lower then threshold, related to the performance of rest of the teams in the system. Agents are free only when the team is dissolved;

Intelligent Recruiting if there are available unemployed agents, recruits agents with capabilities not found in the current team. If there are no unemployed with the desired characteristic, tries to recruit already employed agents. If none of the previous recruiting is successful, then looks for a randomly selected characteristic. The team stops recruiting after a pre-defined number of attempts;

Moderate Risk-Taking Bidding Team takes tasks even not knowing if the task can be finished by the agents currently in the team;

Risk-Taking Bidding Team Strategy tries to take all tasks, even if there are not enough agents in the group. Recruits more agents when necessary;

Specialized Team creates a group with the higher number of agents with the same ability and hires temporary members according to the tasks.

3.2.3 Conclusion

Experiments to test the performance of those different kind of teams showed that stable teams strategies can accomplish more tasks than temporary teams, specially when the tasks have a short expiration time. The stable teams performances are very similar for any of the five strategies.

In the study presented on (RATHOD; JARDINS, 2004), there was no comparison between other methods of group formation nor a study about how the profit values can change the agents decision taking process. The problem was simplified in the way that every agent can perform a task with the same quality and there are no constraints about when an agents is able to join a certain team (for instance, geographic constraints). Thy consider that communication was complete and unrestricted for the team leaders and there are no failures in the information exchange. Those assumptions are not realistic in several domains.

3.3 Clustering

Clustering can be defined as the process of organizing objects into groups whose members are similar in some aspect. A cluster is a collection of elements which are similar within the cluster and are distinct to the elements belonging to other clusters.Usually, the similarity criterion is distance, so two or more objects belong to the same cluster if they

are close according to a given distance, that can be the geometrical distance or some other similarity measure.

Clustering methods usually follow either a hierarchical strategy or one in which observations are relocated among tentative clusters. Hierarchical methods proceed by stages producing a sequence of partitions, each corresponding to a different number of clusters. They can be either *agglomerative* or *divisive*. The agglomerative methods starts with clusters with one element and includes more elements and divisive methods stats with one cluster containing all elements and divides it at each stage. Neither hierarchical nor relocation methods directly address the issue of determining the number of groups within the data. Various strategies for simultaneous determination of the number of clusters and cluster membership have been proposed.

A classical relocation clustering algorithm is called "K-means" (MACQUEEN, 1967). It works by choosing K random points, called centroids, and assigning each data point to the closest centroid. The criterion function is the average squared distance of the data items from their nearest cluster centroids. The centroids are re-calculated to be the center of the formed clusters and the data points are re-assigned using the new centroids. The process is repeated until the centroids do not move. The main advantage of this algorithm is its simplicity. Its disadvantages are that it does not yield the same result with each run and the number of clusters must be given beforehand. Since the resulting clusters depend on the initial random assignment, some clusters may even be left empty if their centroids lie initially far from the distribution of data. The choice of the number of clusters may be critical since quite different kinds of clusters may emerge when K is changed. The pseudo-code for this method presented on Algorithm 3.1.

Algorithm 3.1: K-means

```
1 Place K points (centroids) into the space represented by the elements (objects) that
  are being clustered;
2 repeat
3     foreach element do
4         Assign element to the group that has the closest centroid;
5     Recalculate the positions of the K centroids;
6 until centroids no longer move ;
```

The classical methods consider that the complete information is available and are centralized solutions, they. Centralized clustering is problematical if data is distributed. Decentralization, is a harder problem since even in the centralized case, where each data item can be compared to every other data item, it is hard to find perfect clusters. Decentralization creates the additional problem since even if the correct clusters are found with the available information, the location of items belonging to these clusters also needs to be discovered.

We can use clustering methods for creating groups of agents that share some similarity. In (OGSTON; STEEN; BRAZIER, 2004) the objective was to study a simplified example of agent grouping to gain insight into dynamics that can be used to create self-organizing agent communities. The clustering problem is seen as a search problem in a multi-agent system in which individual agents have the goal of finding similar agents. Agents try to form groups among themselves, and these groups constitute a clustering. As a result each agent will always have a view of only a very small fraction of the rest of the system. They

have created an abstract model of simplified agents which have a very small range of actions and act using straightforward decision functions. Furthermore, these agents can generally only use direct communication with a limited amount of additional coordination among small groups.

Agents are defined by their characteristics and can be considered as a set of data items. Each agent has a small number of communication channels (links) to other agents and define the neighborhood of an agent. The aim of the system is for agents to rearrange these links and to select some of them to form connections or connected links between themselves, generating a graph of connections corresponding to a clustering.

The creation of initial links is derived from the placement of agents and model them as a random network. In the simulations each agent starts out as a cluster of a single item with links to some other randomly chosen agents. As a simulation progresses, agents pick some of their links to become matches, or matched links based on the similarity of the agents joined by the link. Clusters choose the best of these matches proposed by their agents to become connections. Agents joined by a path of connected links form a single cluster.

The creation of connected links allows clusters to expand. They represent the best clusters agents can see in an extremely limited local view. We give agents two behaviors that are used to improve clusters. First, agents in a cluster combine their individual pools of links, widening their individual neighborhoods. This allows them (still individually) to pick better matched links as candidates for cluster membership. Additionally, to prevent agents conglomerating into one large cluster, a limit is placed on cluster size. A further procedure then allows clusters to break weaker connections, enabling them to upgrade stronger available matches into connections. Since connections are between agents, breaking a connection can split a cluster, but leaves other stronger agent pairs connected in the resulting clusters.

The search for good matches between similar agents is represented as a matchmaking problem among agents objectives. Internally, each agent have a main attribute that describes its basic characteristics and are clustered according to these attributes. Each agent further contains a number of objectives, or current goals based on its attribute. Objectives are represented by nearby points, chosen as a function of their agent's attribute. An objective could be only one of many tasks that an agent needs to complete to reach a final goal which is its central attribute. Agents select matches, determining how closely related their objectives are to other agents' objectives. Clusters have a wider view of relative closeness on the attribute level since they contain a larger number of matched and connected links. Clusters have a stronger basis on which to choose connections to make and break. The objective of these agents, with the local coordination provided by cooperation within the clusters, is to have a self-organizing system grouping agents based on their attributes.

The experiments shows that cluster of agents adjust their size to the size of underlying data clusters, and to learn the appropriate range for matching. The agents studied in the Ogston paper exhibit the basic dynamics of forming clusters based on similarities between agent attributes.

3.4 Conclusions

In this chapter we have presented some approaches that can be used when agents need too coordinate their actions in order to perform a common task. The three main methods analyzed have some characteristics, that are summarized on Table 3.4.

Characteristic	**Coalition**	**Cluster**	**Team**
Communication	complete	peer/complete	complete
Leader	not present	not present	present
Solution	centralized/distributed	distributed	semi-centralized
Dynamic formation	yes/no	yes/no	yes/no
Duration	short-therm	short/long-therm	short/long-therm
Reward	distributed	-	distributed
Group value	yes	no	no

As we can see on Table 3.4, the methods have some different requirements, such as communication and centralization. Given these characteristics, we can chose the best method for group-formation where agents need to perform one or more joint actions. On which problem we should use each group formation method? This question is closely related to the problem characteristics, meaning that:

Coalition: problems were the information about the value of the possible coalitions is known;

Clustering: scenarios where the agents are situated and have communication restrictions;

Teams: scenarios where some agents have more access to information or communication (team leaders) and the agents have different abilities.

This simple classification can be used to decide wich method to use to form groups of agents. However, those methods are similar and several grouping problems can be solved using more than one method. In the next chapter we describe two complex domains, and in those domains all three methods can be used.

4 DOMAINS

In this chapter we present the domains on which the proposed approach will be applied. Both domains are challenging for multiagent system and have the desired characteristics, defined on Chapter 1: distributed, dynamic, discrete, nondeterministic, partially observable. We will present their MDP analysis to show that both have a huge state space, the key characteristic that motivates the use of a new approach on both domains. At the end of this Chapter, we present a comparative discussion.

4.1 Urban Traffic Control

Urban traffic control (UTC) is getting each day more complex and traffic congestion is a problem in almost every high-populated urban area. The expansion of the traffic network with the creation of alternative ways is no longer a possible solution in most of cities. Systems to control urban traffic flow are being requested to solve complex scenarios. To achieve an increase of the traffic flow and the decrease of the polluting gases emission, stops, total travel times, among others; is a constrained problem.

The most common way to control traffic is the installation of traffic lights at street junctions. The insertion of a traffic light can solve traffic security problems, but can also generate a decrease of the flow and an increase of the travel time. In order to have a well configured traffic light, it must have a control that determines the *stage* specification, *splits*, *cycle time* and, in coordinated traffic lights, *offset*. Those basic concepts will be presented on the next Subsection. The remaining Subsections will present a non-agent based approach, MAS approaches, MDP analysis of this domain, microscopic simulation, and the application of single agent RL techniques on this domain. A more detailed discussion on UTC and intelligent transportation systems can be found in (BAZZAN; KLÜGL, 2007).

4.1.1 Basic Concepts

Computer technology has been over the years applied to optimize traffic lights timing and create better stage specifications. In general, programs like MAXBAND (LITTLE; KELSON; GARTNER, 1981) and TRANSYT (TRANSYT-7F, 1988) calculate based on historical data, optimal timing for each traffic light, in order minimize some optimization criteria, e.g., the total number of vehicle stops.

Analyzing the scenario in Figure 4.1, it is possible to exemplify some problem characteristics. The *stage* specification is the traffic movements in each part of the cycle time. Figure fig:stages, shows an example of two stages specification for a junction. Each of these stages must have a relative green duration, this share of the cycle time is called *split*.

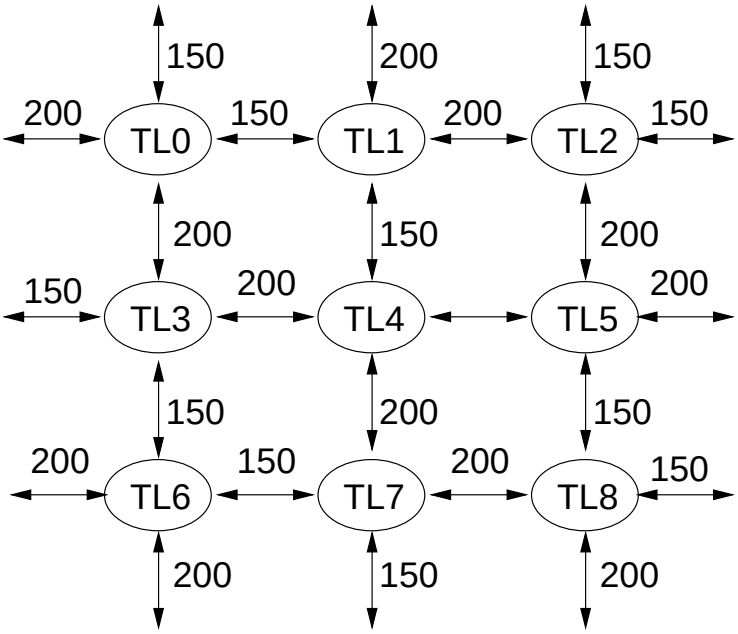


Figure 4.1: Simple scenario

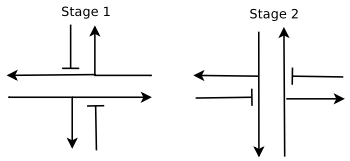


Figure 4.2: Stages movement specification

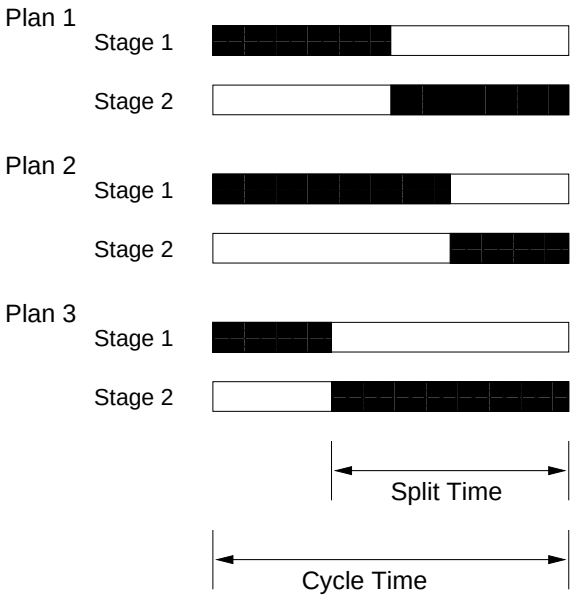


Figure 4.3: Basic signal plans specification

In Figure 4.3 there are three different signal plans specifications, in “Plan 1” both stages have an equal share of the cycle time, in “Plan 2", stage 1 has twice the time than stage 2, and in “Plan 3” stage 2 has twice the time than stage 1.

As an example, consider a vehicle that enters the scenario from North, headed to

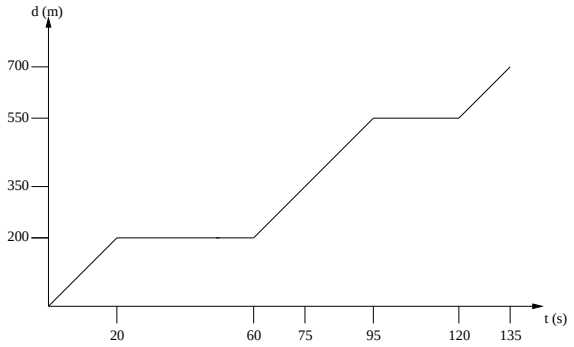


Figure 4.4: Distance x Time

South, must pass through the traffic lights “TL1", “TL4" and “TL7", with average speed of 10m/s. This vehicle would take 70 seconds to cross all the traffic lights without stopping and go out of the scenario. Considering that all traffic lights in this scenario have the same stage specification, seen in Figure 4.2, set to the “Plan 3" and, with a cycle time of 60 seconds. In this conditions, the vehicle would take 20s to get at the “TL1" junction, wait 40s at the red light, wait 25s at “TL7", and end the travel in 135 seconds, this is graphically represented in Figure 4.4. This simple example shows that the configuration of the traffic lights timing can generate a considerable delay time for one or more travel directions in a street.

The control of traffic lights can be isolated or coordinated. The isolated control can be with fixed time, that calculates the splits an cycle times in order to allow the maximum flow in a specific junction, or traffic responsive. Fixed pre-timed plans are developed using manual traffic counts data. Usually, the isolated traffic responsive systems works using inductive loop detectors (ILD), usually located at 40m from the stop line, to determine the queue size stopped at the red light and then change the phase to allow the vehicle to cross the junction. Inductive loop detectors consist of one or more loops of wire embedded in the pavement and connected to a control box, excited by a signal. When a vehicle passes over or rests on the loop, the inductance of the loop is reduced showing the presence of a vehicle.

Isolated intersections control, both fixed or traffic-responsive, is only useful when the traffic condition is undersaturated, (PAPAGEORGIOU et al., 2003). Not always the best timing for one isolated traffic light is the best for the network as a whole. Consider the example scenario, if traffic light “TL4" is giving a larger split time for stage 2 without any *offset* time, the cars might have to wait in the next traffic light (“TL1" or “TL7") and maybe the next traffic light cannot get more traffic then its already receiving, so, an isolated configuration can compromise the performance of the whole network.

Coordinated control means that the traffic lights *offsets* are designed to allow a vehicle that is traveling in a certain average time is able to cross several of junctions controlled by coordinated traffic lights without stopping. Let’s consider again the situation described before, where a vehicle enters the scenario and must cross three different traffic lights. The *offset* time of “TL4" can be relative to the distance between its neighbors, according to the desired synchronization (coordination), and a given average vehicle velocity. To calculate the *offset* time to synchronize with “TL1" is the time that a vehicle takes to cover the distance between both traffic lights with an average velocity, in this situation: $\frac{150m}{10m/s} = 15s$, so, the split start time must be delayed in 15s, in the cycle time, to have a proper coordination between “TL4" and “TL1" in the North-South direction.

Coordinated control can be fixed or traffic responsive, the difference between the two systems is that the fixed control strategies use historical data and the traffic responsive use real-time data. The problem with fixed coordination is that the values are calculated to be optimal for a certain historical data and the traffic patterns can change dramatically during a period of time for many reasons like: traffic adaptation to the calculated times, accidents, natural events, among others. The most used tool for coordination, called TRANSYT, is of this kind.

Traffic responsive coordinated control have a better performance compared to the previous systems, although, they require the installation and operation of communication mechanisms, sensors, and controllers in each traffic light and a central controller. All current applications of this kind, have a centralized method of calculating the optimal timing for each traffic light. One problem with the centralization is that the central process receives a huge amount of data and must distribute the traffic lights configurations for each of the traffic lights in its control, and this can demand some large processing time.

Although there exists many developed traffic control strategies, most of the cities are still operating fixed-time signal control strategies, hard to maintain and to optimize. According to (PAPAGEORGIOU et al., 2003), even when modern traffic-responsive control systems are installed, the employed control strategies are sometimes naïve and poorly tested. Another problem is that traffic responsive systems are usually applied only in some arterials of the network and this might cause problems in the rest of the network, like oversaturating fixed-time controlled streets.

4.1.2 Traffic-responsive Urban Traffic Control (TUC)

Several methodologies look for the traffic flow optimization. However, the majority of those approaches have low performance under saturated conditions (i.e.: SCOOT and SCATS). In other hand, sophisticated tools like OPAC and PRODYN have algorithms with exponential complexity. The TUC (Traffic-responsive Urban Traffic Control) mechanism, (DIAKAKI et al., 2003), was created to provide a traffic responsive UTC method even under saturated conditions.

The TUC strategy consists of four main components: splits control, cycle control, offset timing and priorization of public transportation. Each one of the four components can be used together of individually. The measures of the network links conditions are sent to a local controller at each cycle time. Those measures must be obtained using video cameras or inductive loop detectors (see definition on Subsection 4.1.1). Besides those collected traffic information, it uses information about the public transportation.

Split : The basic methodology for split control is the formulation of the urban traffic control problem as a Linear-Quadratic (LQ) optimal control problem based on a store-and-forward type of mathematical modeling. This part has been described in detail in (DIAKAKI; PAPAGEORGIOU; ABOUDOLAS, 2002), The control objective is to minimize the risk of oversaturation of link queues by varying, in a coordinated manner, the green-phase durations of all stages at all network junctions around some nominal values without affecting neither the offsets nor the cycle times.

Cycle Time : One single cycle time is considered here for a whole network in order to enable junction coordination via suitable offsets. The objective of cycle control should be to increase the junctions capacities through the application of a feedback algorithm that uses as criterion for the increase or decrease of the cycle, the current

maximum saturation level of a pre-specified percentage of the network links. The feedback algorithm for cycle control comprises the following three steps:

1. A pre-specified percentage of network links with maximum load $\sigma_2(k) = \frac{x_2(k)}{X_{c,max}}$ are identified and the corresponding loads are averaged to provide the average maximum load (k).
2. The network cycle is calculated from the feedback control law (P-regulator):

$$C(k) = C^N - K^c(\sigma(k) - \sigma^N) \tag{4.1}$$

   where C^N is a nominal network cycle time, N is a nominal average load, and K^c is a control parameter, the value of which affects the intensity of the control reactions. After applying Equation 4.1, the calculated cycle time is constrained within the minimum and maximum permissible cycle time.
3. If the resulting network cycle time $C(k)$ is sufficiently high while all links approaching specific junctions have sufficiently low saturation levels, these undersaturated junctions run cycle times equal to $C(k)/2$.

Offset The offset control considers the following assumptions:

- Initially specify the offsets along one-directional arterials that do not intersect;
- In case of two-directional arterials, an offset is specified for each direction and the offset that will be finally implemented is a weighted mean of the offsets of the two directions, or, the most loaded direction may be selected (in real time) to determine the arterial offsets;
- In arterials that intersect, considers a pre-specified priority order of the arterials according to their relative importance, and then implement the control to each arterial sequentially from the highest to the lowest priority.

Public Transport Priority when its necessary, coordination is broken at a certain point to answer to a priority request. The coordination is re-implemented after one cycle time.

4.1.2.1 Scenarios and Experiments

In (DIAKAKI et al., 2003), the TUC approach was tested in two real scenarios: Tel Aviv and Jerusalem, depicted on Figure 4.5. In both networks the control of the phase duration is applied once at each traffic lightcycle. The offset control and cycle time changes are activated at every 7.5 minutes.

On both scenarios, TUC was simulated in a 4 hour period, corresponding to the time window from 6 a.m. to 10 a.m., using data based on the morning peak demand expected for the year 2010, when the Light Rail Train (LRT) system will be in place. The performace was compared to a fixed control (including pre-calculated off-sets), developed by traffic engineers using specific designing tools.

When all control modules of TUC where applied in the Tel Aviv network, an average increase of 65% in the mean speed compared to optimal fixed-time control was obtained. In the Jerusalem network, a similar improvement is obtained compared to fixed-time control. The application of all TUC control modules in the Jerusalem network obtained an increase in the the mean speed by 51%, in average.

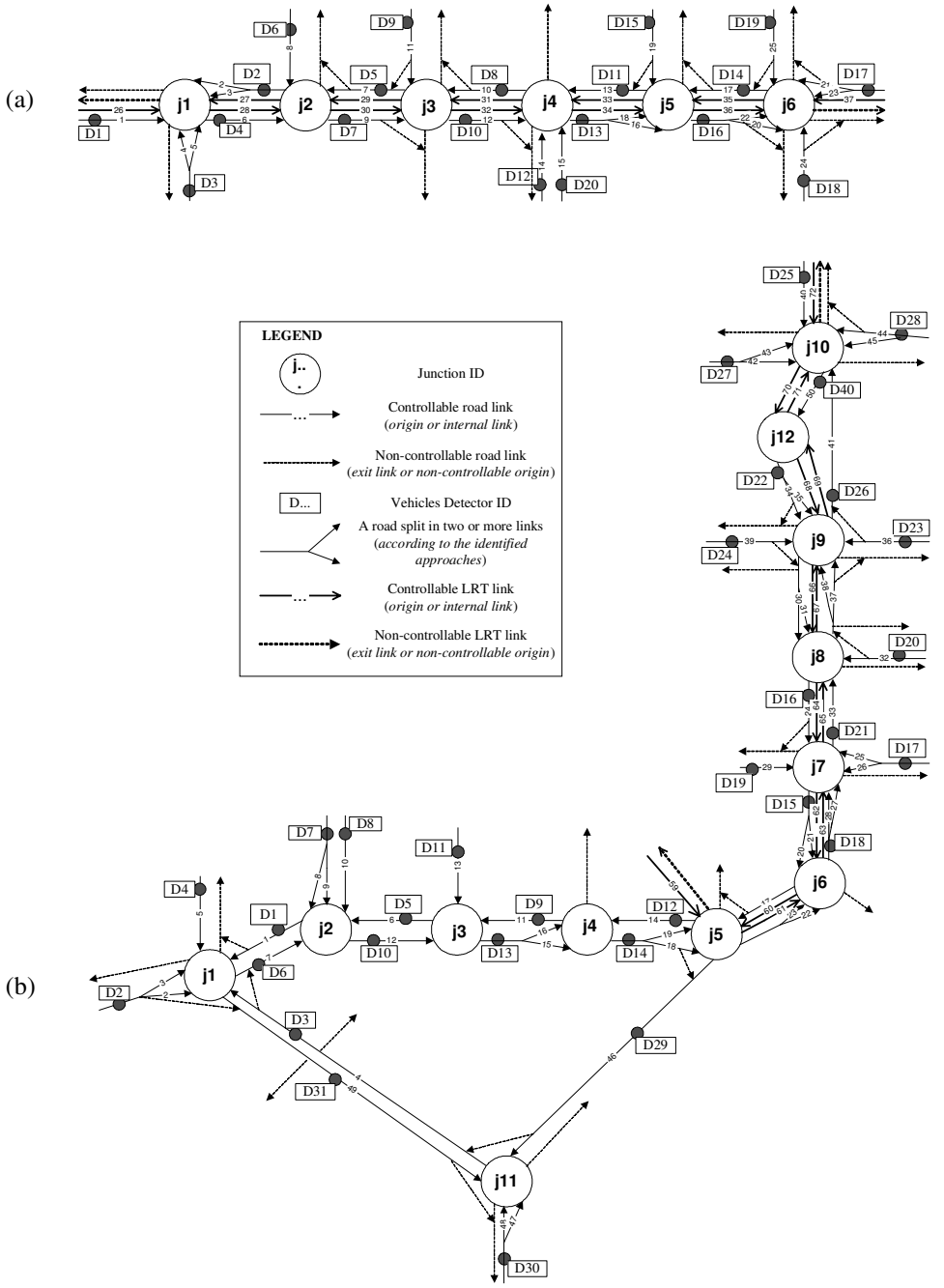


Figure 4.5: Networks: (a) Tel Aviv and (b) Jerusalem

4.1.2.2 Conclusions

The complete problem is solved by a central control that has access to complete information about the traffic on the network. A failure in the central control or in the data transmission can lead to failure in the UTC system. In the presented scenarios, the networks used in the experiments are realtively small and the traffic priorities are known beforehand. This knowledge about the traffic flow behavior eliminates the problem of having different coordination possibilities for a traffic light, so there is no need to have an algorithm to detect critical points and to solve conflicting coordinations.

4.1.3 MAS Approaches

In (BAZZAN, 2005), a MAS based approach is described in which each traffic light is modeled as an agent, each having a set of pre-defined signal plans to coordinate with neighbors. Different signal plans can be selected in order to coordinate in a given traffic direction or during a pre-defined period of the day. This approach uses techniques of evolutionary game theory: self-interested agents receive a reward or a penalty given by the environment. Moreover, each agent possesses only information about their local traffic states. However, payoff matrices (or at least the utilities and preferences of the agents) are required, i.e these figures have to be explicitly formalized by the designer of the system.

In (OLIVEIRA; BAZZAN; LESSER, 2005) an approach based on cooperative mediation is proposed, which is a compromise between totally autonomous coordination with implicit communication and the classical centralized solution. An algorithm to deal with distributed constraint optimization problems (OptAPO) is used in a dynamic scenario, showing that the mediation is able to reduce the frequency of miscoordination between neighbor crossings. However, this mediation was not decentralized: group mediators communicate their decisions to the mediated agents in their groups and these agents just carry out the tasks. Also, the mediation process may take long in highly constrained scenarios, having a negative impact in the coordination mechanism.

In (OLIVEIRA et al., 2004) a decentralized, swarm-based approach was presented, but we have not collected and analyzed information about the group formation. Therefore, a decentralized, swarm-based model of task allocation was developed in (OLIVEIRA; BAZZAN, 2006, 2007), in which the dynamic group formation without mediation combines the advantages of those two previous works (decentralization via swarm intelligence and dynamic group formation).

Camponogara and Kraus (CAMPONOGARA; KRAUS JR., 2003) have studied a simple scenario with only two intersections, using stochastic game-theory and reinforcement learning. Their results with this approach were better than a *best-effort* (greedy), a random policy, and also better than Q-learning.

Finally, approaches based on self-organization of traffic lights via thresholds (GERSHENSON, 2005) or reservation-based systems (DRESNER; STONE, 2004) have still to solve low-level abstraction issues in order to be adopted by traffic engineers and have a chance to be deployed.

4.1.4 MDP Analysis

Modeling the urban traffic control problem as an MDP is essential for solving it using reinforcement learning mechanisms. Mapping the problem, the number of possible states in the system (N) is relative to the number (n) of input links of each traffic light(l_i) and the number of possible states ($|S|$) that those links can have, this number is given by Equation 4.2. The number of possible joint actions ($\mathscr{A}$) is defined by Equation 4.3, where A is the set of possible actions for an agent, considering that on every state the number of possible actions is the same.

$$N = |\times {l_i}^{|S|}| \tag{4.2}$$

$$\mathscr{A} = |\times A^n| \tag{4.3}$$

Using Q-learning, each state-action is an entry in the Q-table, so in this scenario we have a Q-table of size $N \times \mathscr{A}$, the number of possible states multiplied by the number

of possible joint actions. It is easy to realize that even in a small scenario the number o possible Q-values is huge. For instance, consider a scenario where three traffic lights with four input links each, where each input link has three possible states. In this case, the number of possible states is 262,144 ($N = 4^3 \times 4^3 \times 4^3 = 4^9$) and the number of possible joint actions is 27 ($\mathscr{A} = 3^3$), considering that each traffic light has three possible actions. The number of Q-values is 7,077,888 (27×262144).

This simple analysis shows that a centralized solution with reinforcement learning in any scenario with a reasonable number of traffic light is just intractable. However, a MDP framework extension for problems with decentralized control (DEC-DMDP) should also be considered. Agents may have different partial information about the system state and the information held by other agents in the system. For this reason, solving decentralized partially observable MDPs (DEC-POMDPs) is significantly harder, in complexity, than solving a centralized solution.

Considering that the problem of deriving the optimal policy for POMDPs is generally computationally intractable (NEXP-complete), as seen on Chapter 2 Section 2.1. There exist several suboptimal solutions that adopt concepts from game theory or use local optimization techniques, although the quality of their solutions depends on the problem that they are trying to solve.

The distributed urban traffic control problem have some important characteristics to be considered: large number of possible states, limited communication, limited observation, limited action frequency and delayed reward information. We can also include the fact that, modeling the traffic problem as a DEC-POMDP the reward is equal to all agents, meaning that the objective is having a good traffic flow in the network as a whole, not only in main arterials.

4.1.5 Microscopic Simulation

There are two main approaches to the simulation of traffic: macroscopic and microscopic. The microscopic allows the description of each road user as detailed as desired (given computational restrictions), thus permitting a model of drivers' behaviors. Multi-agent simulation is a promising technique for microscopic traffic models as the drivers' behavior can be described incorporating complex and individual decision-making.

In general, microscopic traffic flow model describe the act of driving on a road i.e. the perception and reaction of a driver on a short time-scale. In these models, the drivers are the basic entities and their behavior is described using several different types of mathematical formulations, such as continuous models (e.g. car-following). Other models use cellular automata (CA). In particular, we use the Nagel-Schreckenberg model (NAGEL; SCHRECKENBERG, 1992) because of its simplicity. The Nagel-Schreckenberg cellular automaton model represents a minimal model in the sense that it is capable to reproduce basic features of real traffic.

In the following, the definition of the model for single lane traffic is briefly reviewed. The road is subdivided in cells with a length varying around 7.5 or 5 meters (for highways or urban traffic respectively). Each cell is either empty or occupied by only one vehicle with an integer speed $v_i \in \{0, \ldots, v_{max}\}$, with v_{max} the maximum speed. The motion of the vehicles is described by the following rules (*parallel dynamics*):

R1 Acceleration: $v_i \leftarrow min(v_i + 1, v_max)$,

R2 Deceleration to avoid accidents: $v'_i \leftarrow \min(v_i, gap)$,

R3 Randomizing: with a certain probability p
do $v_i'' \leftarrow \max(v_i' - 1, 0)$,

R4 Movement: $x_i \leftarrow x_i + v_i''$.

The variable gap denotes the number of empty cells in front of the vehicle at cell i. A time-step corresponds to $\Delta t \approx 1$ sec, the typical time a driver needs to react.

Every driver described by the Nagel-Schreckenberg model can be seen as a reactive agent: autonomous, situated in a discrete environment, and has (potentially individual) characteristics such as its maximum speed v_{max}, and the deceleration probability p. During the process of driving, it perceives its distance to the predecessor (gap), its own current speed v, etc. This information is processed using the three rules (**R1-R3**) and changes in the environment are made using rule **R4**. The first rule describes one goal of the agentas it wants to go by maximum speed v_{max}. The other goal is to drive safe i.e. not to collide with its predecessor (**R2**). In this rule the drivers assumes that its predecessor can brake to zero speed. However, this is a very crude approximation of the perception of an agent. These first two rules describe deterministic behavior i.e. the stationary state of the system is determined by the initial conditions. But drivers do not react in this optimal way: they vary their driving behavior without any obvious reasons. This uncertainty in the behavior is reflected by the *braking noise p* (**R3**). It mimics the complex interactions with the other agents i.e. the overreaction in braking and the delayed reaction during acceleration. Finally, the last rule is carried out, the agent acts on the environment and moves according to his current speed.

4.1.6 Applying Single Agent RL on UTC

Traffic optimization is a hard control problem because it deals with highly dynamic and non-stationary flow patterns. Thus, it is important to state beforehand our assumptions regarding the type of dynamics we expect to happen. Specifically, the class of non-stationary environments that we deal with is similar to the one studied by Hidden-Mode MDPs researchers (CHOI; YEUNG; ZHANG, 2001). We assume that the following properties hold: **1)** environmental changes are confined to a small number of *contexts*, which are stationary environments with distinct dynamics; **2)** the current context cannot be directly observed, but can be estimated according to the types of transitions and rewards observed; **3)** environmental context changes are independent of the agent's actions; and **4)** context changes are relatively infrequent.

In a traffic scenario, these assumptions mean that flow patterns are non-stationary but they can be nearly divided in stationary dynamics. We do not assume, however, that this division must be known *a priori*. In fact, one of the interesting aspects of RL-CD is exactly its capability of automatically partitioning the environment dynamics into relevant partial models. Moreover, we assume that the current flow pattern is not given or directly perceivable, but can be estimated by observing attributes such as the queue of cars, street densities, etc.

Our approach relies on RL methods because they provide online learning capabilities and do not rely on offline analysis. The RL-CD mechanism assumes that the use of multiple partial models of the environment is a good approach for dealing with non-stationary scenarios such as traffic control. The use of multiple models makes the learning system capable of partitioning the knowledge in a way that each model automatically assumes for itself the responsibility for "understanding" one kind of flow pattern. To each model, we assign an optimal policy, which is a mapping from traffic conditions to signal plans,

and a trace of prediction error of transitions and rewards, responsible for estimating the quality of a given partial model.

Moreover, the creation of new models is controlled by a continuous evaluation of the prediction errors generated by each partial model. In the following subsections we first describe how to learn contexts (i.e, estimate models for traffic patterns), and then we show how to detect and switch to the most adequate model given a sequence of observations.

All experiments were realized using an implementation of the Nagel-Schreckenberg model, described on Subsection 4.1.5, namely the one in the ITSUMO (SILVA et al., 2006).

4.1.6.1 Scenario and Experiments

Our scenario consists of a traffic network which is a 3x3 Manhattan-like grid, with a traffic light in each junction. Figure 4.6 depicts this network, where the 9 nodes correspond to traffic lights and the 24 edges are directed (one-way) links, each having a maximum allowed velocity. In Figure 4.6, the links depicted in black, dotted, and gray have the maximum allowed velocity set to 54Km/h, 36km/h, and 18Km/h respectively. These velocities correspond to 3, 2, and 1 cell per time step.

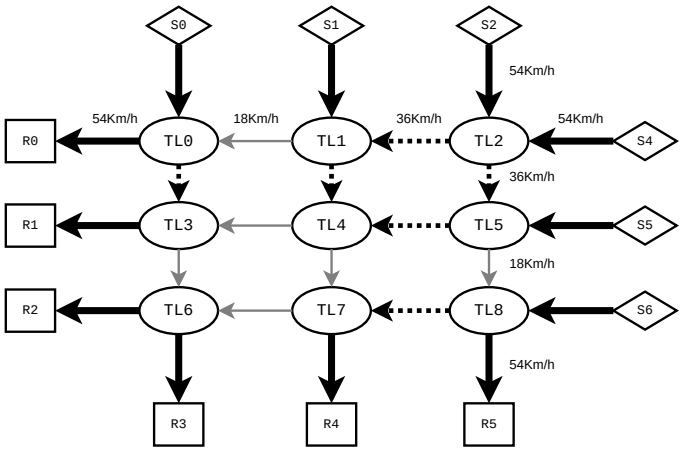


Figure 4.6: A Network of 9 Intersections.

Each link has a capacity of 50 vehicles. Vehicles are inserted by *sources* and removed by *sinks*, depicted as diamonds and squares, respectively, in Figure 4.6. In order to model the non-stationarity of the traffic behavior, our scenario assumes 3 different traffic patterns (contexts). Each consists of a different car insertion distribution. These 3 contexts are:

- *Low*: low insertion rate in the both North and East sources, allowing the traffic network to perform relatively well even if the policies are not optimal (i.e., the network is undersaturated);
- *Vertical*: high insertion rate in the North sources (S0, S1, and S2), and average insertion rate in the East (S3, S4, and S5);
- *Horizontal*: high insertion rate in the East sources (S3, S4, and S5), and average insertion rate in the East (S0, S1, and S2).

Vehicles do not change directions during the simulation and upon arriving at the sinks they are immediately removed. For instance, a vehicle inserted in the network by the source "S0" with South direction will be removed by sink "R3".

We modeled the problem in a way that each traffic light is controlled by one agent, each agent making only local decisions. Even though decisions are local, we assess how well the mechanism is performing by measuring global performance values. By using reinforcement learning to optimize isolated junctions, we implement decentralized controllers and avoid expensive offline processing.

As a measure of effectiveness for the control systems, usually one seeks to optimize a weighted combination of stopped cars and travel time. In our experiments we evaluate the performance by measuring the total number of stopped vehicles, since this is an attribute which can be easily measured by real inductive loop detectors.

After the discretization of the traffic measure, the state of each link can be either *empty*, *regular* or *full*, based on its occupation. The state of an agent is given by the states of the links arriving in its corresponding traffic light. Since there are two one-way links arriving at each traffic light (one from north and one from east), there are 9 possible states *for each agent*. The reward for each agent is given locally by the sum squared of the incoming links' queues. Performance, however, is evaluated for the whole traffic network by summing the queue length of all links, including external queues.

Traffic lights usually have a set of signal plans used for different traffic conditions and/or time of the day. We consider here three plans, each with two phases: one allowing green time to direction north-south (NS), and other to direction east-west (EW). Each one of the three signal plans uses different green times for phases: signal plan **1** gives equal green times for both phases; signal plan **2** gives priority to the vertical direction; and signal plan **3** gives priority to the horizontal direction. All signal plans have cycle time of 60 seconds and phases of either 42, 30 or 18 seconds (70% of cycle time for the preferential direction, 50% of cycle time and 25% of cycle time for non-preferential direction). The signal plan with equal phase times gives 30 seconds for each direction (50% of the cycle time); the signal plan which prioritizes the vertical direction gives 42 seconds to the phase NS and 18 seconds to the phase EW; and the signal plan which prioritizes the horizontal direction gives 42 seconds to the phase EW and 18 seconds to the phase NS.

In our simulation, one time-step consists of two entire cycles of signal plan, that means 120 seconds. The agent's action consists of selecting one of the three signal plans at each simulation step.

In the following experiments we compare the RL-CD against a greedy policy and against classic model-free and a model-based reinforcement learning algorithms. We show that reinforcement learning with context detection performs better than both for the traffic light control problem.

	Sources	
Traffic pattern	S0, S2, S3	S4, S5, S6
Low	0.05	0.05
Vertical	0.20	0.05
Horizontal	0.05	0.20

Table 4.1: Insertion Rates

We change the context (traffic pattern) every 200 observation time steps, which corresponds to 3h and 20min of real traffic flow (12,000 seconds), and each simulation corresponds to 25h (90,000 seconds). The probability of a vehicle being inserted, at each second, during each traffic pattern by the sources is given in Table 4.1. Moreover, all fol-

lowing figures which compare the performance of control methods make use of the metric described in the scenario, that is, the total number of stopped cars in all links (including external queues, that are queues of cars trying to enter the scenario) during the simulation time. This means that the lower the value in the graph, the better the performance.

We have first implemented the greedy policy for the sake of comparison. The greedy solution is a standard decentralized solution for traffic-responsive networks in which there is no coordination. Each agent takes decisions based only on the state of its input links, selecting the signal plan which gives priority to the direction with higher occupation. If the state of both links is the same, then the greedy agent selects the signal plan with equal time distribution.

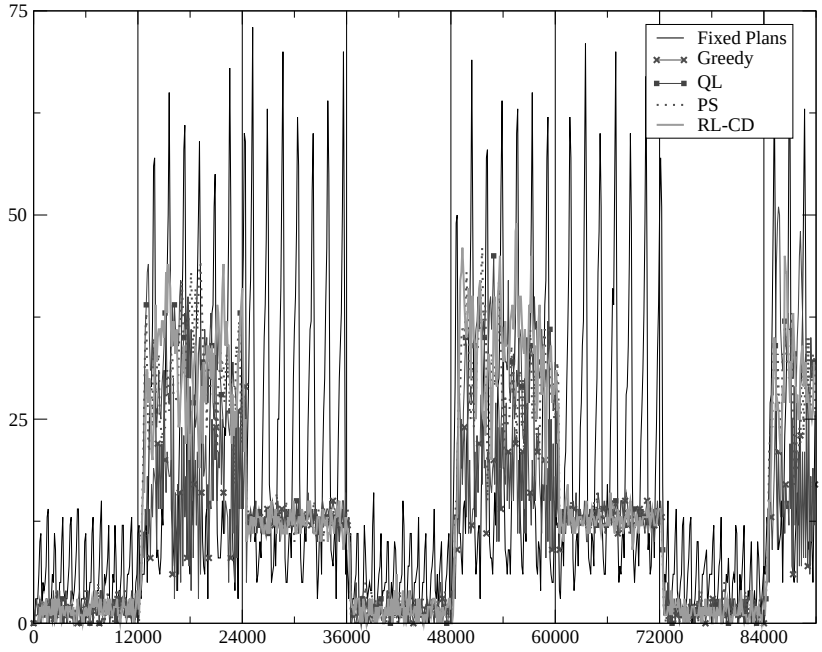


Figure 4.7: Performance of the different control methods in a scenario with no deceleration probability.

Figure 4.7 shows the comparison between RL-CD, fixed signal plans, greedy selection, and two standard RL methods, namely Q-Learning and Prioritized Sweeping, in a scenario where the vehicles have no probability of deceleration which regulates the deceleration behavior of the individual drivers via the parameter p. We can see on this graph that the performance of all learning methods are very similar to the greedy selection mechanism. This shows that in a ideal scenario, with no noise (no deceleration), greedy and the learning methods have a similar performance .

In Figure 4.8 we show the performance of all methods in a scenario where all the vehicles have a probability set to 0.1 of decelerating. In this case, the noise makes the greedy method and the Q-Learning have a lower performance compared to RL-CD and PS.

The results in the other two scenarios, where the deceleration is 0.2 and 0.3, respectively are seen on Figure 4.9 and Figure 4.10. As we can observe, the deceleration p causes the traffic lights to observe the same state (in this case, the high occupation of the input lanes), so that all learning methods are not able to cope with the over saturated scenario. However, for the same reason it is not able to build interesting models of the

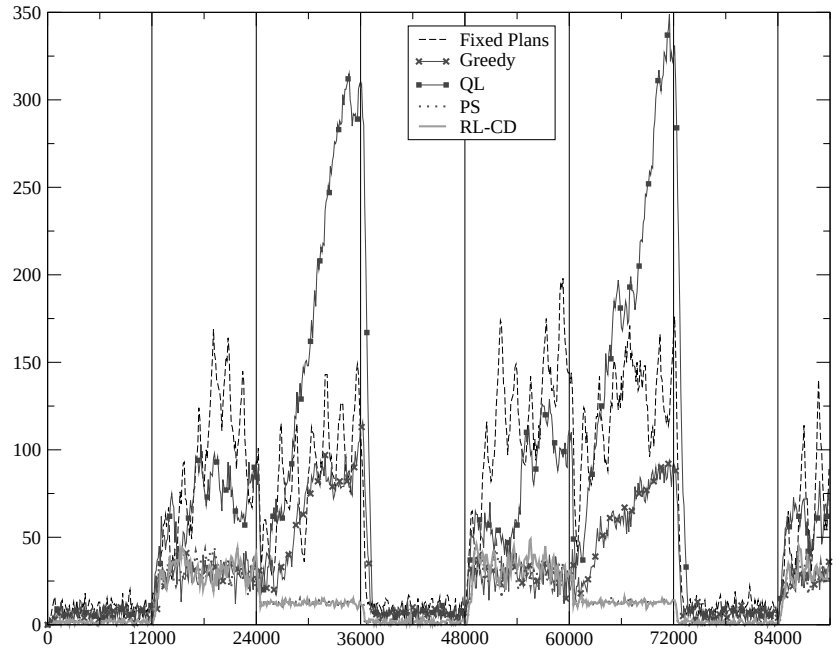


Figure 4.8: Performance of the different control methods in a scenario with deceleration probability set to 0.1.

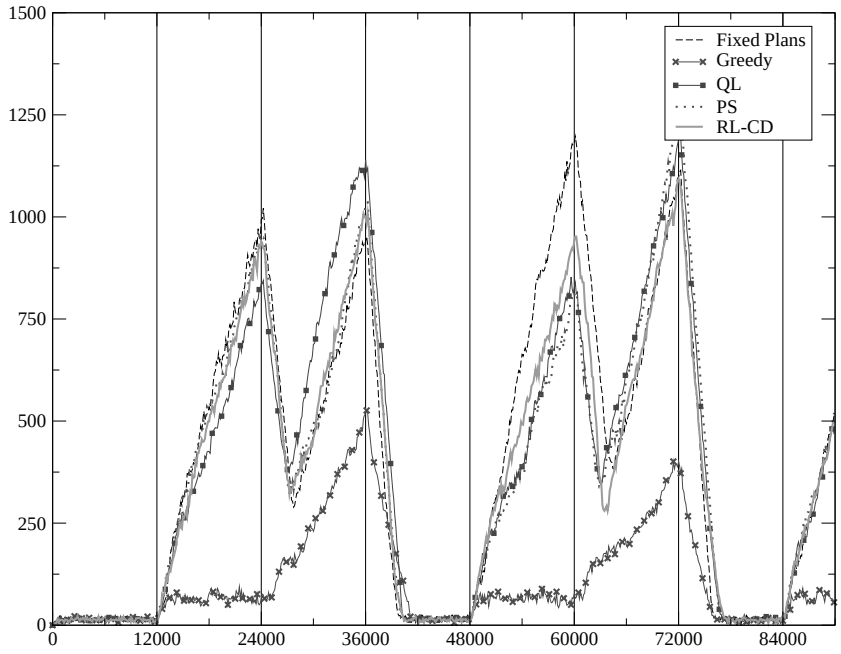


Figure 4.9: Performance of the different control methods in a scenario with deceleration probability set to 0.2.

relevant attributes of the dynamics. Since Q-Learning is model-free, it is less prone to wrong bias caused by non-stationarity. Prioritized Sweeping, on the other hand, tries to build a single model for the environment and ends up with a model which mixes properties of different traffic patterns. For this reason, it can at most calculate a policy which

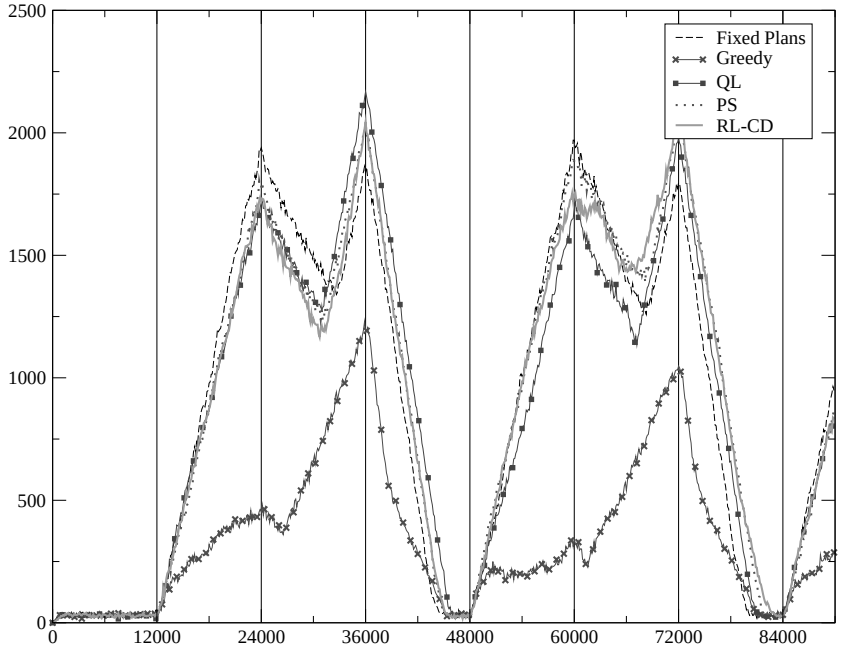


Figure 4.10: Performance of the different control methods in a scenario with deceleration probability set to 0.3.

is a compromise for several different (and sometimes opposite) traffic patterns. The fixed plan and the learning mechanisms have the same performance, indicating that the learning mechanisms are using only the fixed plan that gives the same priority to both directions.

4.1.6.2 Conclusions

In these experiments we have used reinforcement learning methods capable of dealing with non-stationary traffic patterns in a microscopic simulator. This kind of simulator allow the test of several sources of non-stationarity as for instance the deceleration probability, a characteristic of drivers, which is not present in macroscopic models of simulation. The results show that in face of higher degree of non stationarity the learning mechanisms have more difficulty in recognizing and discriminating the states than when the macroscopic model that was used in (SILVA et al., 2006). Also shows that independent learning mechanisms are not capable of dealing with over-saturated networks.

4.2 Urban Search and Rescue

In this section we describe the urban search and rescue (SAR) domain. More precisely the Kitano et al. (KITANO et al., 1999) view of this domain.

4.2.1 RoboCup Rescue

In 1995 a large earthquake hit Kobe City, killing over 6,000 people, injured at least 300,000 and destroyed 103,521 buildings. Immediately after the earthquake, over 300 fires where reported. How this disaster was presented in (KITANO, 2000) as one of the main motivations for creating the urban search and rescue simulator RoboCup Rescue.

The goal of the RoboCup Rescue Simulation is to provide a complex multiagent do-

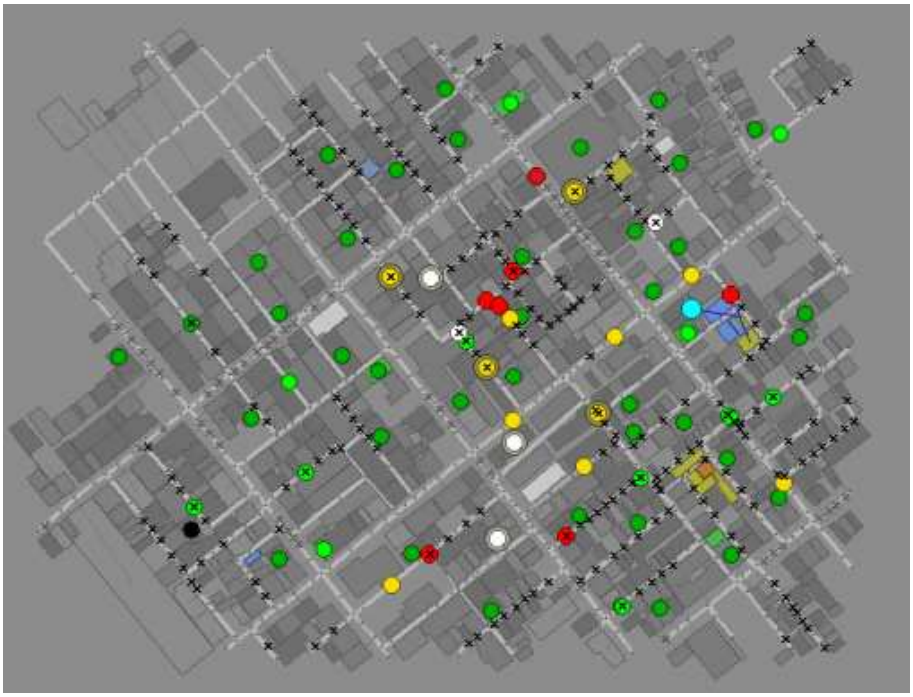


Figure 4.11: RoboCup Rescue 2D Map

main that has some major issues like: heterogeneity, information access, long-term planning and real-time (KITANO et al., 1999). Here multiple fire engines, ambulances and police cars must save civilians from trapped, burning buildings without a global central control to allocate all the resources. Each group with a specific characteristic has a central control (i.e.: Ambulance Center, Fire Station and Police Office) but there is no control that controls all those different centers. For instance, an ambulance must collaborate with a fire engine have a fire extinguished before it can rescue a civilian. A simulation map screeenshot can be seen on Figure 4.11.

Each agent has perception limited to its visual field. Agents are in contact by radio, but they have a limited number of messages that can be exchanged. Direct communication (without radio) among agents in a communication area is unlimited. Therefore, since no agent has a complete information about the global state, the domain is in general collectively partially observable (NAIR et al., 2003).

4.2.2 MDP Analysis

Similar to the analysis made for the UTC domain, here we present a MDP analysis of the SAR, more specifically using some RoboCup Simulation assumptions. There exists three types of controlable active agents: “Fire Brigade” (FB), “Ambulance Team” (AT) and “Police Force” (PF). All agents have four movement actions: "North", "South", "East", and "West". Besides the movement actions the agents have the folowing specific actions:

FB "Extinguish";

AT "Rescue", "Load" and "Unload";

PF "Move" and "Clear".

The number of possible joint actions ($\mathscr{A}$) is given by the number of all combinations of those three agents types and their possible actions, this is given by Equation 4.4.

$$\mathscr{A} = ||4^{n_{FB}}| \times |2^{n_{AT}}| \times |2^{n_{PF}}|| \tag{4.4}$$

where n_{FB}, n_{AT} and n_{PF} are the number of agents of type FB, AT and PF, respectively.

In this scenario we can estimate the total number of states considering the buildings an the streets having only two possible states. Buildings can be on fire or not and streets can be obstructed or free, while each agent can be in one street. Equation 4.5 gives the number of states using those assumptions.

$$N = |2^{n_s} \times 2^{n_b} \times {n_s}^{n_a}| \tag{4.5}$$

where n_s, n_b and n_a are the number of streets, buildings and agents ($n_a = n_{FB} + n_{AT} + n_{PF}$), respectively.

Using Q-learning in this domain, we have a Q-table of size $N \times \mathscr{A}$, the number of possible states multiplied by the number of possible joint actions. It is possible to observe that the size of Q-values is huge for this problem. As as example, consider a scenario where we have 10 agents of each of the categories (FB, AT and OD), 200 streets and 100 buildings. In this case, the number of possible states is $2^{200} \times 2^{100} \times 200^{30}$ and the number of joint actions is 2^{40} ($4^{10} \times 2^{10} \times 2^{10}$), that corresponds to a Q-table of size $2^{200} \times 2^{100} \times 200^{30} \times 2^{40}$. This number is shows that this domains is clearly not solvable using a centralized controller that models the whole environment and the exponential growth is similar to the one observed in the traffic scenario.

4.3 Conclusions

In this chapter we have presented two different domains with similar characteristics, such as: communication restrictions, partial observation, computational complexity and decentralization. Table 4.2 summarizes the characteristics of both domains, showing their similarity.

Table 4.2: Characteristics of SAR and UTC

Feature	SAR	UTC
Number of agents	50 or more	1 to many
Agents in a team	Heterogeneous	Hetero-/Homogeneous
Logistics	Major Issue	No
Long-term Planning	Major Issue	Involved
Emergent Collaboration	Major Issue	Important Issue
Real-Time	Sec. - Min.	Sec. - Min.
Information Access	Very bad	Very bad/Good
Environment	Dynamic	Dynamic
Control	Dist./Semi-central	Dist./Semi-central/Central

The number of agents in the UTC domain is related to the number of traffic lights involved, that is why it can vary from one agent to many. One agent can have two possible

meanings: there is only one intersection controlled by an agent or an agent controls several intersections. In SAR, usually a huge amount of agents is used, since every agent has a different role (i.e: policeman, paramedic, ambulance, etc.).

Consearning teams, both domains have different views of teams. In the SAR domain, a team is a group of agents acting in order to rescue civilians. A team in the UTC scenario is a group trying to have a better traffic flow, usually, in one specific direction ("green-wave"). Collaboration among traffic light agents is restrictive, since an agents that is colaborating to one direction usually is not able to collaborate in other, this restrictive collaboration is no so present in the rescue agents, but can also happen (i.e: two fire brigades trying eliminate the fire from two different building).

In both domains the environment is dynamic, in the sense that the changes are also caused by processes not related to the agents actions. The access to information on the SAR domain is more restricted since the agents are moving and have a limited area of communication. In the UTC local information for a traffic light is easy to obtain (by its detectors) and information about other agents can be obtained by a central control or direct communication.

With the analysis of these domains we perceive that their characteristics are too complex for the current MAS RL approaches. There is a clear need for an approach able to deal with large state space domains, more specifically in domains where a complete representation is not feasible.

5 PROPOSED APPROACH AND APPLICATIONS

In this chapter we present the proposed approach and the applications on the domains presented on Chapter 4.

5.1 Introduction

The objective of this approach is to have a representation of the states where the agent must have more information about the environment or to perform joint actions. The agent starts with no previous knowledge about other agents, not even if there exists any other agent in the scenario. Although, the agent is capable of communicating and observing the environment within a limited observation and communication area. The proposed approach uses communication and negotiation among the agents to have a better view of the environment and to consider joint actions only when those joint actions at certain states are essential. By *essential* we mean that those joint actions are more profitable for the agent than making individual actions at those states. This kind of sparse representation of the states is inspired in the one presented on Chapter 2 at Section 2.4. The sparse and context-specific Q-learning considers the assumption that the agents have information about their coordination dependencies. In the proposed approach, all the coordination mechanisms must not be pre-defined but they shall emerge only from the agents interactions.

As an example, consider a scenario where the agent must move boxes and it is capable of act alone (moving the boxes alone). When the agent is not capable of moving a new box, it will try to obtain information a about the environment to perceive what has changed. There could be something in the box way, not in the agent's observation field, that could be perceived by other agent. In this case, the agent would combine its observation with the observation of the other agent and create a new state, that represents this situation. If the box still not moving, maybe the problem is that more than one agent is needed to move this box, in this case the agent would include incrementally the actions of the agents on the group until there is combination of state and action that represents what is needed to move this kind of box.

It is assumed that the agents have an unlimited communication with the agents in their communication area or group. This kind of cooperation is inspired on the case presented on Chapter 2 at Subsection 2.3.1. In that case, an agent could obtain information about the environment from other agent (partner agent, also called *scout*) in order to have a better representation of the environment and act alone using this information. Here, instead of having a fixed number of possible partners for sharing sensations, the agents must use their perception and try to communicate with the agents in their communication area.

A group formation method is needed in order to have a communication and action group, since the agents will have states where information about other agents is needed.

One of the methods described on Chapter 3 must be chosen at this point. Each of those methods have specific characteristics that makes more suitable for some environments.

In the next section we describe the proposed algorithm.

5.2 Algorithm

Algorithm 5.1: Proposed Approach

$\overrightarrow{s}$ is the current state ;
repeat for each *step of episode*
  Choose $\overrightarrow{a}$ from $\overrightarrow{s}$ using a policy derived from Q ;
  if $\overrightarrow{a} > 1$ **then**
    Negotiate with the agents of $\overrightarrow{a}$;
    if *negotiation failed* **then**
      $\overrightarrow{a} \leftarrow$ my component of $\overrightarrow{a}$;
  Perform action $\overrightarrow{a}$, observe next state s' and reward r ;
  Update Q-table using Procedure 5.2;
  if $E(max_{\overrightarrow{a}'}(\overrightarrow{s}', \overrightarrow{a}')) < E_{min}$ **then**
    $\overrightarrow{s_n} \leftarrow \overrightarrow{s}'$;
    while $|\overrightarrow{s_n}| == |\overrightarrow{s}|$ *or* $|\overrightarrow{s_n}| < |CommunicationArea|$ **do**
      $agent \leftarrow$ NearestAgent $\in$ CommunicationArea ;
      if $state - component\ from\ agent \notin \overrightarrow{s_n}$ **then**
        Add to $\overrightarrow{s_n}$ a $state - component$ from $agent$;
      if $Q(\overrightarrow{s_n}, \overrightarrow{a}') \notin$ *Q-table* **then**
        Add $Q(\overrightarrow{s_n}, \overrightarrow{a}')$ to Q-table;
  else if $|\overrightarrow{s_n}| == |CommunicationArea|$ **then**
    $\overrightarrow{a_n} \leftarrow \overrightarrow{a}'$;
    while $|\overrightarrow{a_n}| == |\overrightarrow{a}'|$ **do**
      $agent \leftarrow$ NearestAgent $\in$ CommunicationArea;
      if $action\ from\ agent \notin \overrightarrow{a_n}$ **then**
        Add to $\overrightarrow{a_n}$ the $action$;
      if $Q(\overrightarrow{s}', \overrightarrow{a_n}) \notin$ *Q-table* **then**
        Add $Q(\overrightarrow{s}', \overrightarrow{a_n})$ to Q-table;
    $\overrightarrow{a}' \leftarrow \overrightarrow{a_n}$;
  $\overrightarrow{s} \leftarrow \overrightarrow{s}'$;
until *s is terminal*;

At the beginning, the agent has only information about the environment states perceived and the set of possible actions in this environment. When the agent perceives that its information about the environment is generating error (the precision $E(s,a)$ is lower than a limit E_{min}), it tries to obtain information from other agent in order to create a new state on its knowledge base (Q-table). At this point a group is needed, since the agent must communicate in order to obtain more information about the environment. When the agents' knowledge about the state of the environment is already expanded and it still

Procedure `updateQTable` (*state vector: s, action vector: a, state vector: s'*)

1 **foreach** *combination i of state-components of s* **do**
2 **foreach** *combination j of action-components of a* **do**
3 **foreach** *combination k of state-components of s'* **do**
4 **if** $Q(i,j) \in Q-table$ **then**
5 $Q(i,j) \leftarrow Q(i,j) + \alpha \left(r\frac{|j|}{|\overrightarrow{a}|} + \gamma max_{a'} Q(k,a') - Q(i,j) \right)$;
6 Update $E(i,j)$;

having significant errors, it starts to create joint actions with the nearest agents. For performing these new joint actions, the agent must negotiate with the agents that are needed to perform the joint action (line 5). This negotiation mechanism must be defined according to the group formation method used to create the communication and action groups.

There is need to study how to make this precision or error function $E(s,a)$ in order to have a value that indicates the low precision only when the value of the pair (s,a) is changing too much for a stationary problem. This value is relative to the number of times that the pair (s,a) was visited. The limit E_{min} is a decreasing function over time, with a lower limit, that can be defined according to the needed precision. E_{min} must always start with the highest possible value, so the agent do not increase its Q-table until it has already experienced acting with its own information for a sufficient time. The decreasing rate for actualizing E_{min} must be defined according to the problem characteristics.

Table 5.1: Complete Transition Description

State	**Action**	**New State**	**Reward for A_1 and A_2**
$< o_1, o_2 >$	$< a_1, a_2 >$	$< o_1, o_2 >$	$+0.1, +0.1$
	$< a_1, a'_2 >$	$< o_1, o'_2 >$	$+0.1, -0.1$
	$< a'_1, a_2 >$	$< o'_1, o_2 >$	$-0.1, +0.1$
	$< a'_1, a'_2 >$	$< o'_1, o'_2 >$	$-0.1, -0.1$
$< o'_1, o_2 >$	$< a_1, a_2 >$	$< o_1, o_2 >$	$+0.1, +0.1$
	$< a_1, a'_2 >$	$< o_1, o'_2 >$	$+0.1, -0.1$
	$< a'_1, a_2 >$	$< o_1, o_2 >$	$+0.1, +0.2$
	$< a'_1, a'_2 >$	$< o'_1, o'_2 >$	$-0.1, -0.1$
$< o_1, o'_2 >$	$< a_1, a_2 >$	$< o_1, o_2 >$	$+0.1, +0.1$
	$< a_1, a'_2 >$	$< o_1, o'_2 >$	$+0.1, -0.1$
	$< a'_1, a_2 >$	$< o'_1, o_2 >$	$-0.1, +0.1$
	$< a'_1, a'_2 >$	$< o'_1, o'_2 >$	$-0.1, -0.1$
$< o'_1, o'_2 >$	$< a_1, a_2 >$	$< o_1, o_2 >$	$+0.1, +0.1$
	$< a_1, a'_2 >$	$< o_1, o'_2 >$	$+0.1, -0.1$
	$< a'_1, a_2 >$	$< o'_1, o_2 >$	$-0.1, +0.1$
	$< a'_1, a'_2 >$	$< o'_1, o'_2 >$	$-0.1, -0.1$

When the agent needs to perform a joint action, it tries to negotiate with the agents involved on the composed action. If the negotiation completely fails, it acts using its best action. The negotiation step is very important, and here we can use any kind of coalition or team formation approach described on Chapter 3. We can choose the more adequate kind of group formation according the problem constraints of communication

and response time.

The pseudo-code of the proposed approach is presented on Algorithm 5.1.

As an example, consider a scenario where Table 5.1 represents the transitions table for a distributed MDP with two agents (A_1 and A_2) where all the transition probabilities are 1 and they receive different rewards. Both agents are able to receive two different observations (o and o') and have two possible actions (a and a'). The index of each observation and each action indicates the agent responsible performing the action or receiving the observation, so "o_2" represents the observation "o" for agent A_2 and "a'_1" represents the action "a'" for agent A_1.

Table 5.2: Independent Transition Description for Agent A_1

State	**Action**	**New State**	**Reward for** A_1
$< o_1 >$	$< a_1 >$	$< o_1 >$	$+0.1$
$< o_1 >$	$< a'_1 >$	$< o'_1 >$	$+0.1$
$< o'_1 >$	$< a_1 >$	$< o_1 >$	$+0.1$
$< o'_1 >$	$< a'_1 >$	$< o'_1 >$ or $< o_1 >$	$+0.1$ or -0.1

Table 5.3: Independent Transition Description for Agent A_2

State	**Action**	**New State**	**Reward for** A_2
$< o_2 >$	$< a_2 >$	$< o_2 >$	$+0.1$
$< o_2 >$	$< a'_2 >$	$< o'_2 >$	$+0.1$
$< o'_2 >$	$< a_2 >$	$< o_2 >$	$+0.1$
$< o'_2 >$	$< a'_2 >$	$< o'_2 >$	$+0.2$ or -0.1

The states, in the complete representation, are composed by the agents' joint observations and the actions are composed by the agents' joint actions. Using an independent single agent RL approach on both agents, they would consider this environment as non stationary since they cannot observe the complete state description nor the effect of the other agent's actions on the system. Agent A_1 would consider unpredictable the transition from $< o'_1 >$ to $< o'_1 >$ or to $< o'_1 >$ when performing action a'_1. The independent transition table for agent A_1 would be represented by Table 5.2 and for agent A_2 by Table 5.3.

Q-table at t=0	Q-table at t=n	Q-table at t=m
$Q(< o_1 >, < a_1 >)$	$Q(< o_1 >, < a_1 >)$	$Q(< o_1 >, < a_1 >)$
$Q(< o_1 >, < a'_1 >)$	$Q(< o_1 >, < a'_1 >)$	$Q(< o_1 >, < a'_1 >)$
$Q(< o'_1 >, < a_1 >)$	$Q(< o'_1 >, < a_1 >)$	$Q(< o'_1 >, < a_1 >)$
$Q(< o'_1 >, < a'_1 >)$	$Q(< o'_1 >, < a'_1 >)$	$Q(< o'_1 >, < a'_1 >)$
	$Q(< o'_1, o_2 >, < a'_1 >)$	$Q(< o'_1, o_2 >, < a'_1 >)$
		$Q(< o'_1, o_2 >, < a'_1, a_2 >)$

Figure 5.1: Q-table changing over time for Agent A_1

To illustrate this example, we have the representation of the Q-table for Agent 1 on Figure 5.1. This figure shows the Q-table changes over time using when the agent is using the proposed algorithm. At the beginning the agent has a Q-table with its possible observations and actions (at t=0). When the agent perceives that the state $< o'_1 >$ has an error, it adds the observation from agent A_2, creating a new state $< o'_1, o_2 >$ (at t=n) and

including this new state on its Q-table. As the agent A_1 continues, it perceives that even this new state continues with error, so it includes the action from the other agent at this state to have a better prediction, so at t=m, agent A_1 includes the Q value with a composed state and a joint action. Using this new Q-table, the agent at state $< o'_1, o_2 >$ will have the possibility to act alone or to use a joint action. The joint action depends on a negotiation with the agent A_2, responsible for performing action a_2. If the negotiation fails, the agent acts using its single action.

5.3 Applications

We have already studied the behavior of independent learning agents interacting in a complex scenario (OLIVEIRA et al., 2006). Those experiments were presented on Chapter 4 on Subsection 4.1.6. The proposed approach will be also applied on the UTC domain. In the second application domain, RoboCup Rescue, it is important to have a more fixed group of reachable agents, since they have no movement restrictions in the scenario but they have a limited communication and perception area.

5.3.1 Learning in UTC

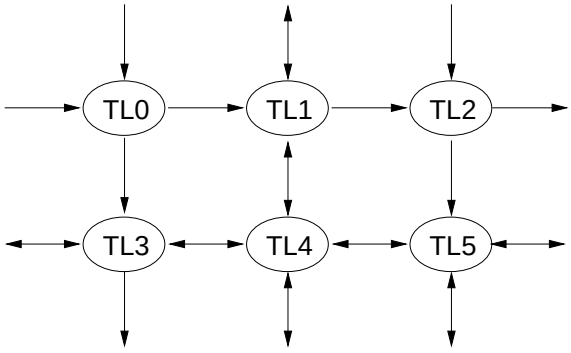


Figure 5.2: Scenario with different communication possibilities.

The agent will act considering their own information about the environment and if it considers the information not sufficient, uses the information received by an another agent. In the traffic scenario, each agent communication area will be restricted according to the number of its incoming links. As an example, consider the scenario on Figure 5.3.1, where the ellipses are representing the traffic light agents and the directed arrows, the links. In this scenario, agent “TL0” has no agents in its communication area, since there are no other agents sending vehicles to its node; agent “TL2” has only one (“TL1”); agents “TL1”,“TL3” and “TL5” have 2; and agent “TL4” has 3 (“TL1”, “TL3” and “TL5”). In this scenario we consider that the agents are always reachable if there are no changes in the topology of the traffic network (i.e:removing one traffic lightor including one new street).

In the UTC scenario, a state with not enough information can be one where based on the local information an action can cause two or more different traffic patterns. The agent can profit from receiving information from its neighboring agents and to have a wider perception of the traffic in some situations, but in others (i.e. low traffic in all links), local information is usually good enough.

Consider the traffic light “TL3” in the map, in a state where all incoming links have high traffic flow, maybe choosing a signal plan that gives no priority to any direction,

some times taking this action makes the agent go to a state where it continues with a high traffic flow on all its links or it changes to a state where on direction now is with a regular flow and the other still over saturated. This situation generates a high value of error at this state, so the agent should combine the observations from the agents in its communication area for solving this issue. In the agents Q-table a new entry is created where the local state is combined with some observations received by its neighbors to have a better prediction of the environment behavior.

Another issue for UTC is when we have coordinated traffic lights forming a “green wave”. In this case, all agents from a given street must have a synchronized timing in order to allow vehicles crossing several junctions without stopping (further explanation on traffic light coordination can be seen on Chapter 4, Section 4.1). In this case, the agents would try to perform the actions, first with its own information, then with the collected data. In the states with a low precision, the agent would include the neighbors actions and have a negotiation in order to achieve a better result.

5.3.2 Learning in the RoboCup Rescue

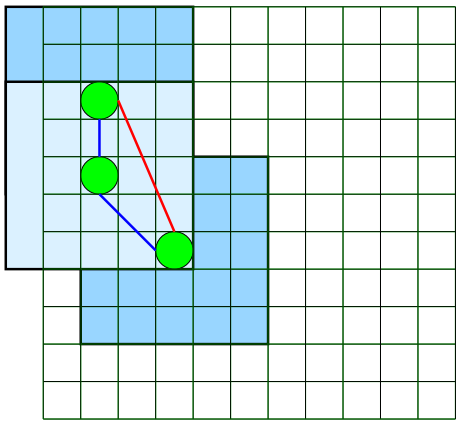
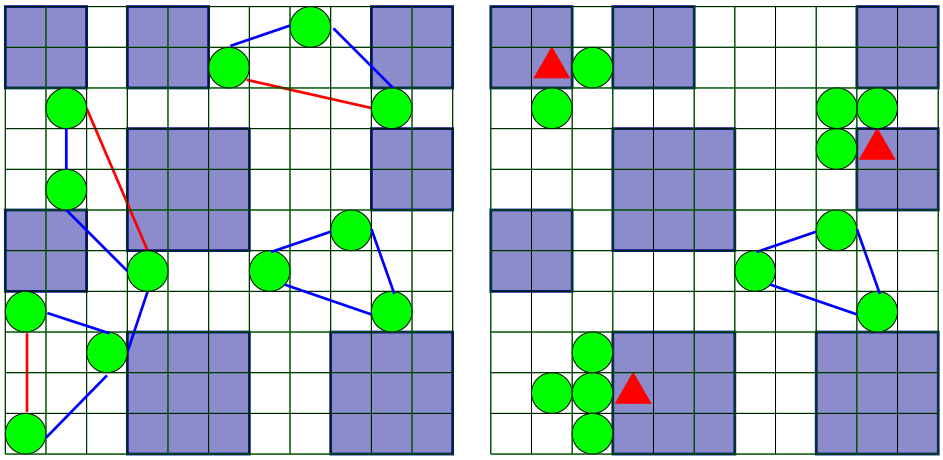


Figure 5.3: Agents communication field.

In this domain, we will focus on learning in a scenario with only Fire Brigade agents. This approach can be use for any of the thee types of agents in RoboCup Rescue but first we will use only one. The use of group formation is a way to have a more permanent set of agents in the communication area, since in this scenario agents are able to change its position. Groups of communication will be formed based on the spatial location of the agents. It is more profitable to form a group with agents in the communication field or near this field. We could also use the communication center present in the RoboCup Rescue simulator, but then the assumption where communication within the group is possible would be broken. Agents will start without any group, forming a stable group according to its communication are, as seen on Figure 5.3.2. Agents can be part of a group if they are able to communicate with at least one of the group members, having only indirect communication with the rest of the group. On Figure 5.4 (a) we can see three formed groups. We can notice that one of the groups has a key member that can break the group in two separated ones. This group formation method is a kind of clustering as seen on Chapter 3 at Section 3.3. Group breaking can be an effect of the tasks that should be performed. For instance, on Figure 5.4 (b) we have three different fire points (triangles), so the large group has broken in two to be able to combat 2 different fire points.



(a) (b)

Figure 5.4: Groups formed (a) and group acting on a joint task (b).

The objective of agent in a group is to learn the joint actions only on when they are needed, using the proposed learning algorithm. In this scenario, a joint action is needed for extinguishing fires. In the other states, agents should learn in a independent way, not considering joint actions, but they can use information received from the agents of its team. The main difference, and advantage, of our proposed approach is that here the agents do not know the cooperative states beforehand. Another important issue groups are formed at the beginning of the simulation based on the spatial perception and communication mechanism of the agents. In this scenario is in the number of possible actions and states that agents acting in the RoboCup Rescue scenario must deal.

Agents should move in formation or have a gathering point to make the formation and to combat the fire. Having a gathering point is better for a wider covering of the map but also increases the number of steps that the agents loose for forming the group and then moving to the fire point. In the RoboCup Rescue scenario, this time can be crucial, since the fires tend to spread very fast and loosing cycles may be crucial.

5.4 Summary

In this chapter we have presented a new approach for multiagent learning, where the agents try to act and learn in the most independent way as possible in the environment before they search for information or actions of other agents. We show the possibilities of using this approach on two complex domains and how its flexible enough to be applied on those scenarios.

6 CONCLUSIONS

There are approaches that lead to the optimal solution on multiagent learning, although it is usually infeasible for problems with many agents mainly due the intractability of the the complete joint action space model. Agents might not be able to observe the state, actions and reward of all other agents and have constrained communication. The domains presented in Chapter 4 are too complex even for a small number of agents if they must be solved using a centralized way or if all the agents have to cooperate on every action. There is a clear need for a framework able to cope with dynamic and complex scenarios, where the agents interact and learn more efficiently.

This work have proposed a new multiagent learning mechanism able to cope with dynamic and distributed decision making problems. In this new approach the agent starts using a simple individual representation and expand it to a more complex environment representation, allowing the agent to increase its perception of the environment and have a better performance.

In the next section the activities that need to be fulfilled in order to implement and test the approaches are presented. The last section contains the activities schedule.

6.1 Activities Plan

The following activities must be fulfilled:

1. Implement the use of information exchange among learning agents (expansion of states part of the approach) and run experiments on the microscopic traffic simulator.

2. Implement the movement and spatial group formation on RoboCup Rescue and run experiments.

3. Implement the complete approach and run experiments on the microscopic traffic simulator.

4. Implement and run experiments using the proposed approach on the RoboCup Rescue simulator, for one kind of agent.

5. Papers to be submitted:

   (a) AAMAS 2008: Independent vs. cooperative learning on the traffic scenario, using the expansion of states of the proposed approach.

   (b) IROS 2008: Spatial group formation on RoboCup rescue.

(c) Journal paper: the complete approach with the achieved results applied to both scenarios must be submitted to one of the following journals: Autonomous Agents and Multi-Agent Systems, Machine Learning Research or Journal of Artificial Intelligence and Research.

6. Redaction of Dissertation

7. Thesis Presentation

6.2 Schedule

The activities will be performed according to the schedule on Tables 6.1, 6.2, and 6.3.

Table 6.1: Schedule for 2007–2.

Activ.	July	August	September	October	November	December
1	•	•	•			
5a		•	•	•		
2				•		
5b					•	•
6	•	•	•	•	•	•

Table 6.2: Schedule for 2008–1.

Activ.	January	February	March	April	May	June
3	•	•	•			
4			•	•	•	
5c					•	•
6	•	•	•	•	•	•

Table 6.3: Schedule for 2008–2.

Activ.	July	August	September	October	November	December
5c	•	•				
6	•	•	•	•		
7					•	

REFERENCES

BAZZAN, A. L. C. A Distributed Approach for Coordination of Traffic Signal Agents. **Autonomous Agents and Multiagent Systems**, [S.l.], v.10, n.1, p.131–164, March 2005.

BAZZAN, A. L. C.; KLÜGL, F. Sistemas Inteligentes de Transporte e Tráfego: uma abordagem de tecnologia da informação. In: KOWALTOWSKI, T.; BREITMAN, K. K. (Ed.). **Anais das Jornadas de Atualização em Informática**. [S.l.]: SBC, 2007.

BECKER, R.; ZILBERSTEIN, S.; LESSER, V. R.; GOLDMAN, C. V. Transition-independent decentralized markov decision processes. In: AAMAS, 2003. **Anais...** New York: ACM, 2003. p.41–48.

BERNSTEIN, D. S.; GIVAN, R.; IMMERMAN, N.; ZILBERSTEIN, S. The Complexity of Decentralized Control of Markov Decision Processes. **Mathematics of Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.27, n.4, p.819–840, 2002.

CAMPONOGARA, E.; KRAUS JR., W. Distributed Learning Agents in Urban Traffic Control. In: EPIA, 2003. **Anais...** [S.l.: s.n.], 2003. p.324–335.

CHALKIADAKIS, G.; BOUTILIER, C. Coordination in Multiagent Reinforcement Learning: a bayesian approach. In: SECOND INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS), 2003. **Proceedings...** [S.l.: s.n.], 2003. p.709–716.

CHALKIADAKIS, G.; BOUTILIER, C. Bayesian Reinforcement Learning for Coalition Formation under Uncertainty. In: AAMAS, 2004. **Anais...** [S.l.: s.n.], 2004. p.1090–1097.

CHALKIADAKIS, G.; BOUTILIER, C. Coalitional Bargaining with Agent Type Uncertainty. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 20., 2007. **Proceedings...** [S.l.: s.n.], 2007. p.1227–1232.

CHOI, S. P. M.; YEUNG, D.-Y.; ZHANG, N. L. Hidden-Mode Markov Decision Processes for Nonstationary Sequential Decision Making. In: SUN, R.; GILES, C. L. (Ed.). **Sequence Learning**: paradigms, algorithms, and applications. Berlin: Springer, 2001. p.264–287.

CLAUS, C.; BOUTILIER, C. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: FIFTEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.746–752.

DIAKAKI, C.; DINOPOULOU, V.; ABOUDOLAS, K.; PAPAGEORGIOU, M.; BEN-SHABAT, E.; SEIDER, E.; LEIBOV, A. Extensions and New Applications of the Traffic Signal Control Strategy TUC. In: ANNUAL MEETING OF THE TRANSPORTATION RESEARCH BOARD, 82., 2003. **Proceedings...** [S.l.: s.n.], 2003. p.12–16.

DIAKAKI, C.; PAPAGEORGIOU, M.; ABOUDOLAS, K. A Multivariable Regulator Approach to Traffic-Responsive Network-Wide Signal Control. **Control Engineering Practice**, [S.l.], v.10, n.2, p.183–195, February 2002.

DOYA, K.; SAMEJIMA, K.; KATAGIRI, K.; KAWATO, M. Multiple Model-Based Reinforcement Learning. **Neural Computation**, Cambridge, MA, USA, v.14, n.6, p.1347–1369, 2002.

DRESNER, K.; STONE, P. Multiagent Traffic Management: a reservation-based intersection control mechanism. In: THE THIRD INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 2004, New York, USA. **Anais...** New York: IEEE Computer Society, 2004. p.530–537.

GERSHENSON, C. Self-Organizing Traffic Lights. **Complex Systems**, [S.l.], v.16, n.1, p.29–53, 2005.

GORDON, G. J. Agendas for multi-agent learning. **Artificial Intelligence**, Essex, UK, v.171, n.7, p.392–401, May 2007.

GUESTRIN, C.; LAGOUDAKIS, M. G.; PARR, R. Coordinated Reinforcement Learning. In: NINETEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 2002, San Francisco, CA, USA. **Proceedings...** Morgan Kaufmann, 2002. p.227–234.

HANSEN, E. A.; BERNSTEIN, D. S.; ZILBERSTEIN, S. Dynamic Programming for Partially Observable Stochastic Games. In: NINETEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, SIXTEENTH CONFERENCE ON INNOVATIVE APPLICATIONS OF ARTIFICIAL INTELLIGENCE, 2004. **Proceedings...** AAAI Press / The MIT Press, 2004. p.709–715.

KAELBLING, L. P.; LITTMAN, M. L.; CASSANDRA, A. R. Planning and Acting in Partially Observable Stochastic Domains. **Artificial Intelligence**, [S.l.], v.101, n.1–2, p.99–134, 1998.

KAELBLING, L. P.; LITTMAN, M.; MOORE, A. Reinforcement Learning: a survey. **Journal of Artificial Intelligence Research**, [S.l.], v.4, p.237–285, 1996.

KITANO, H. RoboCup Rescue: a grand challenge for multi-agent systems. In: INTERNATIONAL CONFERENCE ON MULTIAGENT SYSTEMS, 4., 2000, Boston, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.5–12.

KITANO, H.; TADOKORO, S.; NODA, I.; MATSUBARA, H.; TAKAHASHI, T.; SHINJOU, A.; SHIMADA, S. RoboCup Rescue: search and rescue in large-scale disasters as adomain for autonomous agents research. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 1999, Tokyo, Japan. **Proceedings...** [S.l.: s.n.], 1999. v.6, p.739–743.

KOK, J. R. **Coordination and Learning in Cooperative Multiagent Systems**. 2006. Tese (Doutorado em Ciência da Computação) — Faculty of Science, University of Amsterdam.

KOK, J. R.; VLASSIS, N. Sparse cooperative Q-learning. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 21., 2004, New York, USA. **Proceedings...** ACM Press, 2004. p.481–488.

KOK, J.; VLASSIS, N. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. **Journal of Machine Learning Research**, [S.l.], v.7, p.1789–1828, 2006.

LI, L.; WALSH, T. J.; LITTMAN, M. L. Towards a Unified Theory of State Abstraction for MDPs. In: NINTH INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND MATHEMATICS, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.531–539.

LITTLE, J. D. C.; KELSON, M. D.; GARTNER, N. H. **MAXBAND**: a versatile program for setting signals on arteries and triangular networks. [S.l.]: Massachusetts Institute of Technology (MIT), Sloan School of Management, 1981. Working papers. (1185-81).

MACQUEEN, J. B. Some methods of classification and analysis of multivariate observations. In: FIFTH BERKELEY SYMPOSIUM ON MATHEMTICAL STATISTICS AND PROBABILITY, 1967. **Proceedings...** [S.l.: s.n.], 1967. p.281–297.

NAGEL, K.; SCHRECKENBERG, M. A Cellular Automaton Model for Freeway Traffic. **Journal de Physique I**, [S.l.], v.2, p.2221, 1992.

NAIR, R.; TAMBE, M.; YOKOO, M.; PYNADATH, D. V.; MARSELLA, S. Taming Decentralized POMDPs: towards efficient policy computation for multiagent settings. In: EIGHTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI-03), 2003, Acapulco, Mexico. **Proceedings...** Morgan Kaufmann, 2003. p.705–711.

OGSTON, E.; STEEN, M.; BRAZIER, F. Group Formation Among Decentralized Autonomous Agents. **Applied Artificial Intelligence**, [S.l.], v.18, n.9, p.953–970, 2004.

OLIVEIRA, D.; BAZZAN, A. L. C. Traffic Lights Control with Adaptive Group Formation Based on Swarm Intelligence. In: INTERNATIONAL WORKSHOP ON ANT COLONY OPTIMIZATION AND SWARM INTELLIGENCE, ANTS 2006, 5., 2006, Berlin. **Proceedings...** Springer, 2006. p.520–521. (Lecture Notes in Computer Science).

OLIVEIRA, D.; BAZZAN, A. L. C.; LESSER, V. Using Cooperative Mediation to Coordinate Traffic Lights: a case study. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTI AGENT SYSTEMS (AAMAS), 4., 2005. **Proceedings...** New York: IEEE Computer Society, 2005. p.463–470.

OLIVEIRA, D.; BAZZAN, A. L. C.; SILVA, B. C.; BASSO, E. W.; NUNES, L.; ROSSETTI, R. J. F.; OLIVEIRA, E. C.; SILVA, R.; LAMB, L. C. Reinforcement learning based control of traffic lights in non-stationary environments: a case study in a microscopic simulator. In: EUROPEAN WORKSHOP ON MULTI-AGENT SYSTEMS, (EUMAS06), 4., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.31–42.

OLIVEIRA, D. d.; BAZZAN, A. L. C. Swarm Intelligence Applied to Traffic Lights Group Formation. In: VI ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL (ENIA), 2007. **Anais...** SBC, 2007. p.1003–1112.

OLIVEIRA, D.; FERREIRA, P.; BAZZAN, A. L. C.; KLÜGL, F. A Swarm-based Approach for Selection of Signal Plans in Urban Scenarios. In: FOURTH INTERNATIONAL WORKSHOP ON ANT COLONY OPTIMIZATION AND SWARM INTELLIGENCE - ANTS 2004, 2004, Berlin, Germany. **Proceedings...** [S.l.: s.n.], 2004. p.416–417. (Lecture Notes in Computer Science, v.3172).

PANAIT, L.; LUKE, S. Cooperative Multi-Agent Learning: the state of the art. **Autonomous Agents and Multi-Agent Systems**, Hingham, MA, USA, v.11, n.3, p.387–434, 2005.

PAPAGEORGIOU, M.; DIAKAKI, C.; DINOPOULOU, V.; KOTSIALOS, A.; WANG, Y. Review of Road Traffic Control Strategies. **Proceedings of the IEEE**, [S.l.], v.91, n.12, p.2043–2067, December 2003.

PUTERMAN, M. L. **Markov Decision Processes**: discrete stochastic dynamic programming. New York, NY, USA: Wiley–Interscience, 2005. (Wiley Series in Probability and Statistics).

RATHOD, P.; JARDINS, M. des. Stable Team Formation Among Self-Interested Agents. In: AAAI WORKSHOP ON FORMING AND MAINTAING COALITIONS IN ADAPTIVE MULTIAGENT SYSTEMS, 2004. **Proceedings...** Menlo Park: USA: AAAI, 2004. p.29–36.

SANDHOLM, T. Perspectives on Multiagent Learning. **Artificial Intelligence**, Essex, UK, v.171, n.7, p.382–391, May 2007.

SHEHORY, O. Coalition Formation: towards feasible solutions. **Fundamenta Informaticae**, Amsterdam, The Netherlands, v.63, n.2–3, p.107–124, 2004.

SHOHAM, Y.; POWERS, R.; GRENAGER, T. If multi-agent learning is the answer, what is the question? **Artificial Intelligence**, Essex, UK, v.171, n.7, p.365–377, May 2007.

SILVA, B. C. d.; BASSO, E. W.; BAZZAN, A. L. C.; ENGEL, P. M. Dealing with Non-Stationary Environments using Context Detection. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING ICML, 23., 2006. **Proceedings...** New York: ACM Press, 2006. p.217–224.

SILVA, B. C. d.; JUNGES, R.; OLIVEIRA, D.; BAZZAN, A. L. C. ITSUMO: an intelligent transportation system for urban mobility. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS 2006) - DEMONSTRATION TRACK, 5., 2006. **Proceedings...** ACM Press, 2006. p.1471–1472.

SILVA, B. C.; OLIVEIRA, D. d.; BAZZAN, A. L. C.; BASSO, E. W. Adaptive Traffic Control with Reinforcement Learning. In: WORKSHOP ON AGENTS IN TRAFFIC AND TRANSPORTATION (AAMAS 2006), 4., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.80–86.

SOH, L.-K.; LI, X. Adaptive, Confidence-Based Multiagent Negotiation Strategy. In: THIRD INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS), 2004, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.1048–1055.

STONE, P. Multiagent learning is not the answer. It is the question. **Artificial Intelligence**, Essex, UK, v.171, n.7, p.402–405, May 2007.

SUTTON, R.; BARTO, A. **Reinforcement Learning**: an introduction. Cambridge, MA: MIT Press, 1998.

TAN, M. Multi-Agent Reinforcement Learning: independent vs. cooperative agents. In: TENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML 1993), 1993. **Proceedings...** Morgan Kaufmann, 1993. p.330–337.

TOSIC, P. T.; AGHA, G. A. Maximal Clique Based Distributed Coalition Formation for Task Allocation in Large-Scale Multi-agent Systems. In: FIRST INTERNATIONAL WORKSHOP ON MASSIVELY MULTI-AGENT SYSTEMS I (MMAS), 2005, Berlin. **Anais...** Springer, 2005. p.104–120. (LNAI, v.3446).

TRANSYT-7F. **TRANSYT-7F User's Manual**. [S.l.]: Transportation Research Center, University of Florida, 1988.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine Learning**, Hingham, MA, USA, v.8, n.3, p.279–292, 1992.

YOUNG, H. P. The possible and the impossible in multi-agent learning. **Artificial Intelligence**, Essex, UK, v.171, n.7, p.429–433, May 2007.