An Introduction to Multi-Agent Systems

Chapter · July 2010

DOI: 10.1007/978-3-642-14435-6_1

2 authors:

Balaji Parasumanna Gokulan
Huawei Technologies
13 PUBLICATIONS **855** CITATIONS
SEE PROFILE

D. Srinivasan
National University of Singapore
349 PUBLICATIONS **16,821** CITATIONS
SEE PROFILE

Chapter 1

An Introduction to Multi-Agent Systems

P.G. Balaji and D. Srinivasan

Department of Electrical and Computer Engineering
National university of Singapore
g0501086@nus.edu.sg, dipti@nus.edu.sg

Summary. Multi-agent systems is a subfield of Distributed Artificial Intelligence that has experienced rapid growth because of the flexibility and the intelligence available solve distributed problems. In this chapter, a brief survey of multi-agent systems has been presented. These encompass different attributes such as architecture, communication, coordination strategies, decision making and learning abilities. The goal of this chapter is to provide a quick reference to assist in the design of multi-agent systems and to highlight the merit and demerits of the existing methods.

Keywords: Multi-agent systems, Agent architecture, Coordination strategies and MAS communication.

1 Distributed Artificial Intelligence (DAI)

Distributed artificial intelligence (DAI) is a subfield of Artificial Intelligence [1] that has gained considerable importance due to its ability to solve complex real-world problems. The primary focus of research in the field of distributed artificial intelligence has included three different areas. These are parallel AI, Distributed problem solving(DPS) and Multi-agent systems (MAS). Parallel AI primarily refers to methodologies used to facilitate classical AI [2-8] techniques when applied to distributed hardware architectures like multiprocessor or cluster based computing. The main aim of parallel AI is to increase the speed of operation and to work on parallel threads in order to arrive at a global solution for a particular problem. Distributed problem solving is similar to parallel AI and considers how a problem can be solved by sharing the resources and knowledge between large number of cooperating modules known as Computing entity. In distributed problem solving, communication between computing entities, quantity of information shared are pre-determined and embedded in design of computing entity. Distributed problem solving is rigid due to the embedded strategies and consequently offers little or no flexibility.

In contrast to distributed problem solving, Multi-agent systems (MAS) [9-11] deal with the behaviour of the computing entities available to solve a given problem. In a multi-agent system each computing entity is referred to as an agent. MAS can be defined as a network of individual agents that share knowledge and communicate with each other in order to solve a problem that is beyond the scope of a single agent. It is imperative to understand the characteristics of the individual agent or computing entity to distinguish a simple distributed system and multi-agent system.

The chapter is organized into nine sections. Section 2 gives a brief overview of an agent and its properties. The characteristics of multi-agent system is given in section 3. Section 4 shows the general classification of MAS based on their organization and structure. Section 5 gives details of various mechanisms used in the communication of information between agents. Section 6 gives details of the decision making strategies used in MAS and is followed by the coordination principles in section 7. Section 8 gives an insight into the learning process in multi-agent systems,. The advantages and their disadvantages are highlighted. These are followed by section 9 which contains some of the concluding remarks.

2 Agent

Researchers in the field of artificial intelligence have so far failed to agree on a consensus definition of the word "Agent". The first and foremost reason for this is due to the universality of the word Agent. It cannot be owned by a single community. Secondly, the agents can be present in many physical forms which vary from robots to computer networks. Thirdly, the application domain of the agent is vastly varied and it is impossible to generalize. Researchers have used terms like softbots (software agents), knowbots (Knowledge agents), taskbots (task-based agents) based on the application domain where the agents were employed [12]. The most agreed definition of agent was that of Russell and Norvig. They define an agent as a flexible autonomous entity capable of perceiving the environment through the sensors connected to it. These act on the environment through actuators. The definition provided does not cover the entire range of characteristics that an agent should possess. It can be distinguished from expert systems and distributed controllers. Some important traits that differentiate an agent from simple controllers are as follows.

Situatedness: This refers to the interaction of an agent with the environment through the use of sensors and the resultant actions of the actuators. Environment in which an agent is present is an integral part of its design. All of the inputs are received directly as a consequence of the agents interactions with its environment. The agent's directly act upon the environment through the actuators and do not serve merely as a meta level advisor. This attribute makes differentiates it from expert systems in which the decision making node or entity suggests for changes through a middle agent and does not directly influence the environment.

Autonomy: This can be defined as the ability of an agent to choose its actions independently without external intervention by other agents in the network (case of multi-agent systems) or human interference. These attribute protects the internal states of agent from external influence. It isolates the agent from instability caused by external disturbances.

Inferential capability: The ability of an agent to work on abstract goal specifications such as deducing an observation by generalizing the information. This could be done by utilizing relevant contents of available information.

Responsiveness: The ability to perceive the condition of environment and respond to it in a timely fashion to take account of any changes in the environment. This latter property is of critical importance in real-time applications.

Pro-activeness: Agent must exhibit a good response to opportunistic behaviour. This is in order to enhance actions that are goal-directed rather than just being responsive to a specific change in environment. It must have the ability to adapt to any changes in the dynamic environment.

Social behaviour: Even though the agent's decision must be free from external intervention, it must still be able to interact with the external sources when the need arises to achieve a specific goal. It must also be able to share this knowledge and help other agents (MAS) solve a specific problem. That is agents must be able to learn from the experience of other communicating entities which may be human, other agents in the network or statistical controllers.

Some other properties that are associated with the agents include mobility, temporal continuity, collaborative behaviour etc. Based on whether a computing entity is able to satisfy all or a few of the above properties , agents could be further specified as exhibiting either weak or a strong agency.

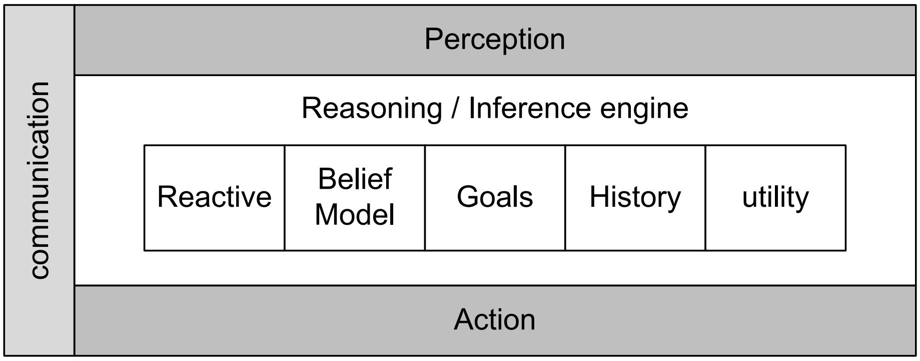


Fig. 1. Typical building blocks of an autonomous agent

It is however extremely difficult to characterize agents based only on these properties. It must also be based on the complexity involved in the design, the function which is to be performed and rationality which is exhibited.

3 Multi-Agent Systems

A Multi-Agent System (MAS) is an extension of the agent technology where a group of loosely connected autonomous agents act in an environment to achieve a common goal. This is done either by cooperating or competing, sharing or not sharing knowledge with each other.

Multi-agent systems have been widely adopted in many application domains because of the beneficial advantages offered. Some of the benefits available by using MAS technology in large systems [13] are

1. An increase in the speed and efficiency of the operation due to parallel computation and asynchronous operation
2. A graceful degradation of the system when one or more of the agent fail. It thereby increases the reliability and robustness of the system
3. Scalability and flexibility- Agents can be added as and when necessary
4. Reduced cost- This is because individual agents cost much less than a centralized architecture
5. Reusability-Agents have a modular structure and they can be easily replaced in other systems or be upgraded more easily than a monolithic system

Though multi-agent systems have features that are more beneficial than single agent systems, they also present some critical challenges. Some of the challenges are highlighted in the following section.

Environment: In a multi-agent system, the action of an agent not only modifies its own environment but also that of its neighbours. This necessitates that each agent must predict the action of the other agents in order to decide the optimal action that would be goal directed. This type of concurrent learning could result in non-stable behaviour and can possibly cause chaos. The problem is further complicated, if the environment is dynamic. Then each agent needs to differentiate between the effects caused due to other agent actions and variations in environment itself.

Perception: In a distributed multi-agent system, the agents are scattered all over the environment. Each agent has a limited sensing capability because of the limited range and coverage of the sensors connected to it. This limits the view available to each of the agents in the environment. Therefore decisions based on the partial observations made by each of the agents could be sub-optimal and achieving a global solution by this means becomes intractable.

Abstraction: In agent system, it is assumed that an agent knows its entire action space and mapping of the state space to action space could be done by experience. In MAS, every agent does not experience all of the states. To create a map, it must be able to learn from the experience of other agents with similar capabilities or decision making powers. In the case of cooperating agents with similar goals, this can be done easily by creating communication between the agents. In case of competing agents it is not possible to share the information as each of the agents tries to increase its own chance of winning. It is therefore essential to quantify how much of the local information and the capabilities of other agent must be known to create an improved modelling of the environment.

Conflict resolution: Conflicts stem from the lack of global view available to each of the agents. An action selected by an agent to modify a specific internal state may be bad for another agent. Under these circumstances, information on the constraints, action preferences and goal priorities of agents must be shared between to improve cooperation. A major problem is knowing when to communicate this information and to which of the agents.

Inference: A single agent system inference could be easily drawn by mapping the State Space to the Action Space based on trial and error methods. However in MAS, this is difficult as the environment is being modified by multiple agents that may or may not be interacting with each other. Further, the MAS might consists of heterogeneous agents, that is agents having different goals and capabilities. These may be not cooperating and competing with each other. Identifying a suitable inference mechanism in accordance of the capabilities of each agent is crucial in achieving global optimal solution.

It is not necessary to use multi-agent systems for all applications. Some specific application domains which may require interaction with different people or organizations having conflicting or common goals can be able to utilize the advantages presented by MAS in its design.

4 Classification of Multi Agent System

The classification of MAS is a difficult task as it can be done based on several different attributes such as Architecture [14], Learning [15][16][17], Communication [14], Coordination [18]. A general classification encompassing most of these features is shown in figure 2.

4.1 Internal Architecture

Based on the internal architecture of the particular individual agents forming the multi-agent system, it may be classified as two types:

1. Homogeneous structure
2. Heterogeneous structure

a) Homogeneous Structure

In a homogeneous architecture, all agents forming the multi-agent system have the same internal architecture. Internal architecture refers to the Local Goals, Sensor Capabilities, Internal states, Inference Mechanism and Possible Actions [19]. The difference between the agents is its physical location and the part of the environment where the action is done. Each agent receives an input from different parts of the environment. There may be overlap in the sensor inputs received. In a typical distributed environment, overlap of sensory inputs is rarely present [20].

b) Heterogeneous Structure

In a heterogeneous architecture, the agents may differ in ability, structure and functionality [21]. Based on the dynamics of the environment and the location of the particular agent, the actions chosen by agent might differ from the agent located in a different part but it will have the same functionality. Heterogeneous architecture helps to make modelling applications much closer to real-world [22].Each agent can have different local goals that may contradict the objective of other agents. A typical example of this can be seen in the Predator-Prey game [23]. Here both the prey and the predator can be modelled as agents. The objectives of the two agents are likely to be in direct contradiction one to the other.

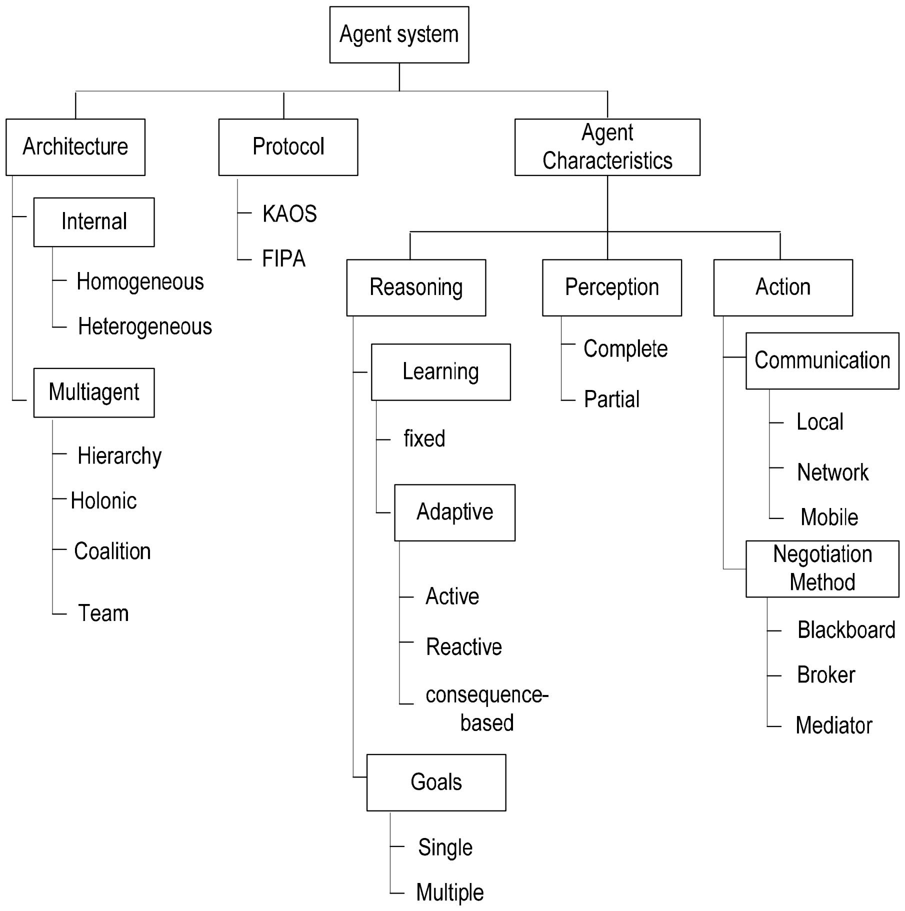


Fig. 2. Classification of a multi agent system based on the use of different attributes

4.2 Overall Agent Organization

a) Hierarchical Organization

Hierarchical Organization [24] is one of the earliest organizational design in multi-agent systems. Hierarchical architecture has been applied to a large number of distributed problems. In the hierarchical agent architecture, the agents are arranged in a typical tree like structure. The agents at different levels on the tree structure have different levels of autonomy. The data from the lower levels of hierarchy typically flow upwards to agents with a higher hierarchy. The Control Signal or Supervisory Signals flow from higher to a lower hierarchy [25]. Figure.3 shows a typical Three Hierarchical Multi-Agent Architecture. The flow of control signals is from a higher to lower priority agents.

According to the distribution of the control between the agents, hierarchical architecture can be further classified as being a simple and uniform hierarchy.

Simple Hierarchy: In a simple hierarchy [26], the decision making authority is bestowed using a single agent of highest level of the hierarchy. The problem with a simple hierarchy is that a single point failure of the agent in the highest hierarchy may cause the entire system to fail.

Uniform Hierarchy: In a uniform hierarchy, the authority is distributed among the various agents in order to increase the efficiency, fault tolerance having a graceful degradation in case of single and multi point failures. Decisions are made by agents having the appropriate information. These decisions are sent up the hierarchy only where there is a conflict of interest between agents in different hierarchy.

Reference [25], provides an example of a uniform hierarchical multi-agent system applied to a urban traffic signal control problem. The objective is to provide a distributed control of traffic signal timings. This is to reduce the total delay time experienced by vehicles in a road network. A Three Level Hierarchical Multi-Agent System where each intersection is modelled as an agent forming the agents at lowest hierarchy followed by zonal agents which supervise a group of lower level agents and finally a single apex supervisor agent at the top of hierarchy. The agent in the lower level of the hierarchy decides on the optimal green time. This is based on the local information collected at each of the intersections. The agents at the higher level of the hierarchy modify decision of the lower hierarchical agents. From time to time there may be a conflict of interest or the overall delay experienced at a group of intersections increases due to a selected action. Here, the overall control is uniformly distributed among the agents. A disadvantage is that the amount and the type of information which must be transmitted to agents at higher hierarchy. This is a non-trivial problem which increases as the network size increases.

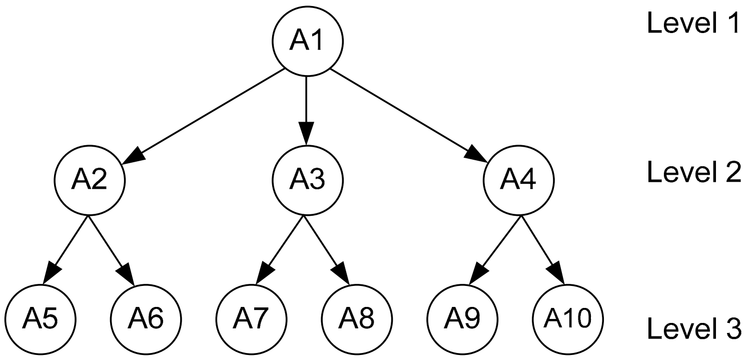


Fig. 3. A Hierarchical Agent Architecture

b) Holonic Agent Organization

A 'Holon' is a stable and coherent similar or fractal structure that consists of several 'holons' as its sub-structure and is itself a part of a larger framework. The concept of a holon was proposed by Arthur Koestler [27] to explain the social behaviour of biological species. However, the hierarchical structure of the holon and its interactions have been used to model a large organizational behaviours in manufacturing and business domains [28][29].

In a holonic multi-agent system, an agent that appears as a single entity to may be composed of many sub-agents bound together by commitments. The sub-agents are not bound by a hard constraints or by pre-defined rule but through commitments. These refer to the relationships agreed to by all of the participating agents inside the holon.

Each holon appoints or selects a Head Agent that can communicate with the environment or with other agents located in the environment. The selection of the head agent is usually based on the resource availability, communication capability and the internal architecture of each agent. In a homogeneous multi-agent system, the selection can be random and a rotation policy could be employed similar to that used with distributed wireless sensor networks. In the heterogeneous architecture, the selection is based on the capability of the agent. The holons formed may group further in accordance to benefits foreseen in forming a coherent structure. They form Superholons. Figure 4. shows a Superholon formed by grouping two holons. Agents A1 and A4 are the heads of the holons and communicate with agent A7. This is the head of the superholon. The architecture appears to be similar to that of hierarchical organization. However in holonic architecture, cross tree interactions and overlapping or agents forming part of two different holons are allowed.

In recent times, [30] had proved the superiority of the holonic multi-agent organization and how the autonomy of the agents increases when in a holonic group. The abstraction of the internal working of holons provides an increased degree of freedom when selecting the behaviour. A major disadvantage[30-31] is the lack of a model or of a knowledge of the internal architecture of the holons. This makes it difficult for other agents to predict the resulting actions of the holons.

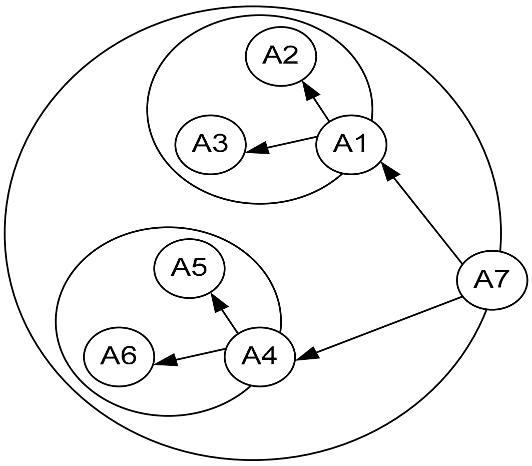


Fig. 4. An example of Superholon with Nested Holons resembling the Hierarchical MAS

c) Coalitions

In coalition architecture, a group of agents come together for a short time to increase the utility or performance of the individual agents in a group. The coalition ceases to exist when the performance goal is achieved. Figure 5. shows a typical coalition multiagent system. The agents forming the coalition may have either a flat or a hierarchical architecture. Even when using a flat architecture, it is possible to have a leading agent to act as a representative of the coalition group. The overlap of agents among coalition groups is allowed as this increases the common knowledge within the coalition group. It helps to build a belief model. However the use of overlap increases the complexity of computation of the negotiation strategy. Coalition is difficult to maintain in a dynamic environment due to the shift in the performance of group. It may be necessary to regroup agents in order to maximize system performance.

Theoretically, forming a single group consisting of all the agents in the environment will maximize the performance of the system. This is because each agent has access to all of the information and resources necessary to calculate the condition for optimal action.It is impractical to form such a coalition due to restraints on the communication and resources.

The number of coalition groups created must be minimized in order to reduce the cost associated with creating and dissolving a coalition group. The group formation may be pre-defined based on a threshold set for performance measure or alternatively could be evolved online.

In reference [32], a coalition multi-agent architecture for urban traffic signal control was mentioned. Each intersection was modelled as an agent with capability to decide the optimal green time required for that intersection. A distributed neuro-fuzzy inference engine was used to compute the level of cooperation required and the agents which must be grouped together.

The coalition groups reorganize and regroup dynamically with respect to the changing traffic input pattern. The disadvantage is the increased computational complexity involved in creating ensembles or coalition groups. The coalition MAS may have a better short term performance than the other agent architectures [33].

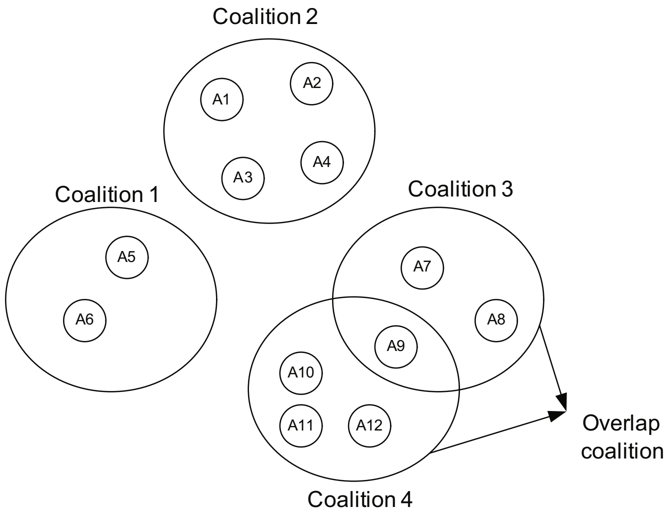


Fig. 5. Coalition multi-agent architecture using overlapping groups

d) Teams

Team MAS architecture [34] is similar to coalition architecture in design except that the agents in a team work together to increase the overall performance of the group. Rather than each working as individual agents. The interactions of the agents within a team can be quite arbitrary and the goals or the roles assigned to each of the agents can vary with time based on improvements resulting from the team performance. Reference [35] , deals with a team based multi-agent architecture having a partially observable environment. In other words, teams that cannot communicate with each other has been proposed for the Arthur's bar problem. Each team decides on whether to attend a bar by means of predictions based on the previous behavioural pattern and the crowd level experienced which is the reward or the utility received associated with the specific period of time. Based on the observations made in [35], it can be concluded that a large team size is not beneficial under all conditions. Consequently some compromise must be made between the amount of information, number of agents in the team and the learning capabilities of the agents.

Large teams offer a better visibility of the environment and larger amount of relevant information. However, learning or incorporating the experiences of individual agents into a single framework team is affected. A smaller team size offers faster learning possibilities but result in sub-optimal performance due to a limited view of the environment. Tradeoffs between learning and performance need to be made in the selection of the optimal team size. This increases the computational cost much greater than that experienced in coalition multi-agent system architecture. Figure 6. shows a typical team based on architecture with partial view. The team 1 and 3 can see each other but not teams 2 ,4 and vice versa. The internal behaviour of the agents and their roles are arbitrary and vary with teams even in homogeneous agent structure.

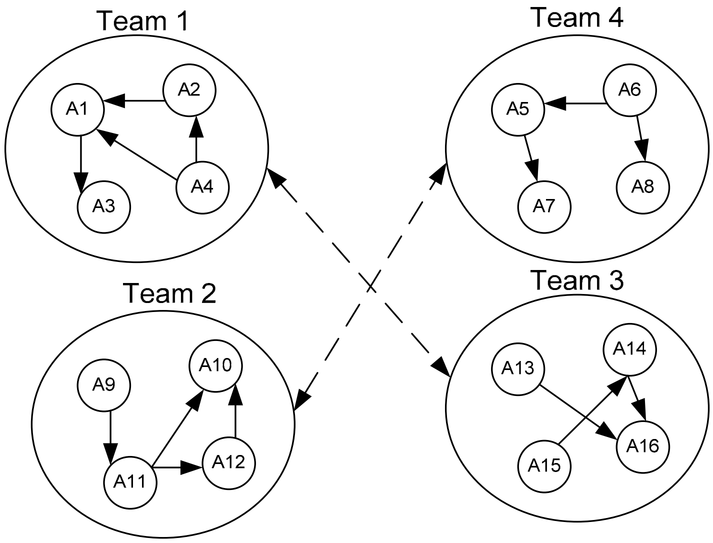


Fig. 6. Team based multi-agent architecture with a partial view of the other teams

Variations and constraints on aspects of the four agent architecture mentioned before can produce other architectures such as federations, societies and congregations. Most of these architectures are inspired by behavioural patterns in governments, institutions and large industrial organizations. A detailed description of these architectures, their formation and characteristics may be found in [34].

5 Communication in Multi-Agent System

Communication is one of the crucial components in multi-agent systems that needs careful consideration. Unnecessary or redundant intra-agent communication can increase the cost and cause instability. Communication in a multi-agent system can be classified as two types. This is based on the architecture of the agent system and the type of information which is to be communicated between the agents. In [14], the various issues arising in MAS system with homogeneous and heterogeneous architecture has been considered and explained by using a predator/prey and by the use of robotic soccer games. Based on the information communication between the agents [36], MAS can classified as local communication or message passing and network communication or Blackboard. Mobile communication can be categorized into class of local communication.

5.1 Local Communication

Local communication has no place to store the information and there is no intermediate communication media present to act as a facilitator. The term message passing is used to emphasize the direct communication between the agents. Figure 7. shows the structure of the message passing communication between agents. In this type of communication, the information flow is bidirectional. It creates a distributed architecture and it reduces the bottleneck caused by failure of central agents. This type of communication has been used in [25] [37] [38].

5.2 Blackboards

Another way of exchanging information between agents is through Blackboards [39]. Agent-based blackboards, like federation systems, use grouping to manage the interactions between agents. There are significant differences between the federation agent architecture and the blackboard communication.

In blackboard communication, a group of agents share a data repository which is provided for efficient storage and retrieval of data actively shared between the agents. The repository can hold both the design data as well as the control knowledge that can be accessed by the agents. The type of data that can be accessed by an agent can be controlled through the use of a control shell. This acts as a network interface that notifies the agent when relevant data is available in the repository. The control shell can be programmed to establish different types of coordination among the agents. Neither the agent groups nor the individual agents in the group need to be physically located near the blackboards. It is possible to establish communication between various groups by remote interface communication. The major issue is due to the failure of blackboards. This could render the group of agents useless depending on the specific blackboard. However, it is possible to establish some redundancy and share resources between various blackboards. Figure 8a. shows a single blackboard with the group of agents associated with it. Figure 8b. shows blackboard communication between two different agent groups and also the facilitator agents present in each group.

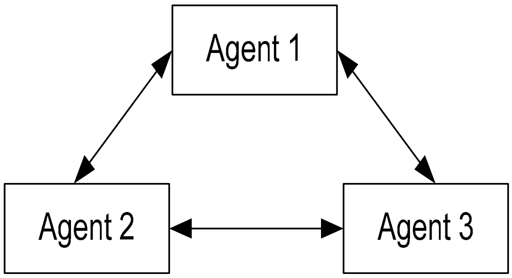
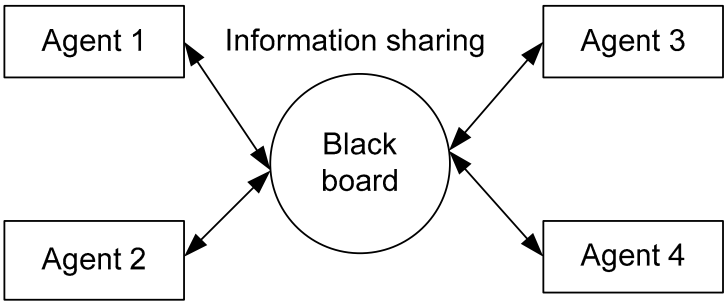
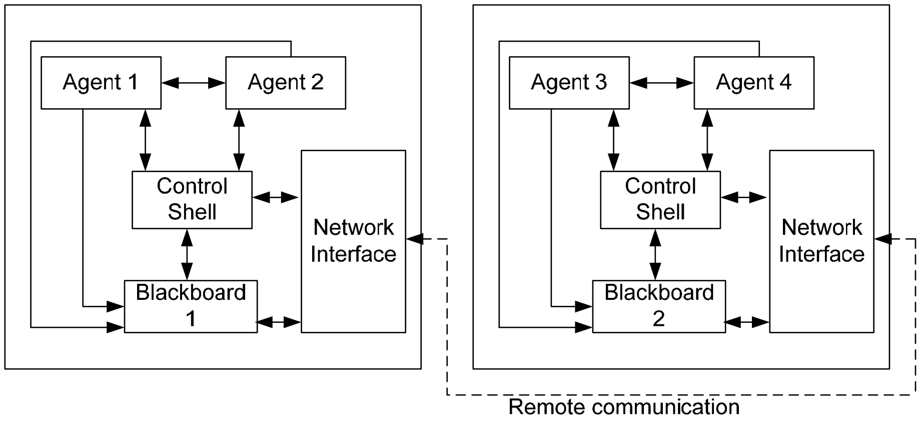


Fig. 7. Message Passing Communication between agents



(a)



(b)

Fig. 8. (a) Blackboard type communication between agents. (b) Blackboard communication using remote communication between agent groups.

5.3 Agent Communication Language

An increase in the number of agents and the heterogeneity of the group necessitates a common framework to help in proper interaction and information sharing. This common framework is provided by the agent communication languages (ACL). The elements that are of prime importance in the design of ACL were highlighted in [40]. They are the availability of the following.

- A common language and interaction format (Protocol) that can be understood by all of the participating agents.
- A shared Ontology where the message communicated has the same meaning in all contexts and follows agent independent semantics.

There are two popular approaches in the design of an agent communication language. They are Procedural approach and Declarative approach. In Procedural approach, the communication between the agents is modelled as a sharing of the procedural directives. Procedural directives shared could be a part of how the specific agents does a specific task or the entire working of the agent itself. Scripting languages are commonly used in the procedural approach. Some of the most common scripting languages employed are JAVA, TCL, Applescript and Telescript. The major disadvantage of the procedural approach is the necessity of providing information on the recipient agent which in most cases is not known or only partially known. In case of making a wrong model assumption, the procedural approach may have a destructive effect on the performance of the agents. The second major concern is the merging of shared procedural scripts into a single large executable relevant script for the agent. Owing to these disadvantages, the procedural approach is not the preferred method for designing agent communication language.

In the declarative approach, the agent communication language is designed and based on the sharing of the declarative statements that specifies definitions, assumptions, assertions, axioms etc. For the proper design of an ACL using a declarative approach, the declarative statements must be sufficiently expressive to encompass the use of a wide-variety of information. This would increase the scope of the agent system and also avoid the necessity of using specialized methods to pass certain functions. The declarative statements must be short and precise as to increase in the length affects the cost of communication and also the probability of information corruption. The declarative statements also needs to be simple enough to avoid the use of a high level language. This means that the use of the language is not required to interpret the message passed. To meet all of the above requirements of the declarative approach based ACL, the ARPA knowledge sharing effort has devised an agent communication language to satisfy all requirements.

The ACL designed consists of three parts [41]: A Vocabulary part, "Inner language" and "Outer language". The Inner language is responsible for the translation of the communication information into a logical form that is understood by all agents. There is still no consensus on a single language and many inner language representations like KIF (Knowledge Interchange Format)[42], KRSL, LOOM are available. The linguistic representation created by these inner languages are concise, unambiguous and context-dependent. The receivers must derive from them the original logical form. For each linguistic representation, ACL maintains a large vocabulary repository. A good ACL maintains this repository open-ended so that modifications and additions can be made to include increased functionality. The repository must also maintain multiple ontology's and its usage will depends on the application domain.

Knowledge Interchange Format [43] is one of the best known inner languages and it is an extension of the First-Order Predicate Calculus (FOPC). Some of the information that can be encoded using KIF are simple data, constraints, negations, disjunctions, rules, meta-level information that aids in the final decision process. It is not possible to use just the KIF for information exchange as much implicit information needs to be embedded. This is so that the receiving agent can interpret it with a minimal knowledge of the sender's structure. This is difficult to achieve as the packet size grows with the increase in embedded information. To overcome this bottleneck, a high level language that utilizes the inner language as its backbone were introduced. These high-level languages make the information exchange independent of the content syntax and ontology. One well known Outer language that satisfies this category is the KQML (Knowledge Query and Manipulation Language) [44]. A typical information exchange between two agents utilizing the KQML and KIF agent communication language is as follows.

```
(ask  :Content (geolocation lax(?long ?lat))
          : language KIF
         :ontology STD_GEO
        : from location_agent
       : to  location_server
      : label Query- "Query identifier")
(tell : content "geolocation(lax, [55.33,45.56])"
     : language standard_prolog
    : ontology STD_GEO)
```

The KQML is conceived as both message format and message handling protocol to facilitate smooth communication between agents. From the above example provided, it can be seen that KQML consists of three layers (Figure 9): A communication layer which indicates the origin and destination agent information and query label or identifier, a message layer that specifies the function to be performed (eg: In the example provided, the first agent asks for the geographic location and the second agent replies to the query), and a content layer to provide the necessary details to perform the specific query.

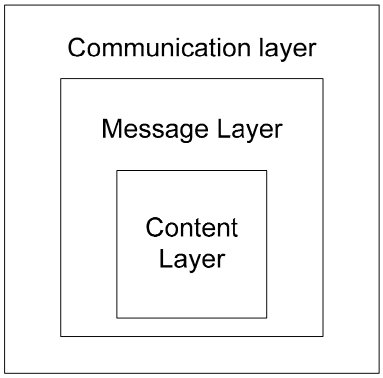


Fig. 9. KQML - Layered language structure

In KQML, the communication layer is at a low level and packet oriented. A stream oriented approach is yet to be developed. The communication streams could be built on TCP/IP, RDP, UDP or any other packet communication media. The content layer specifies the language to be employed by the agent. It should be noted that agents can use different languages to communicate with each other and interpretation can be performed locally by higher level languages.

6 Decision Making in Multi-Agent System

Multi-agent decision making is different from a simple single agent decision system. The uncertainty associated with the effects of a specific action on the environment and the dynamic variation in the environment as a result of the action of other agents makes multi-agent decision making a difficult task. Usually the decision making in MAS is considered as a methodology to find a joint action or the equilibrium point which maximizes the reward received by every agent participating in decision making process. The decision making in MAS can be typically modelled as a game theoretic method. Strategic game is the most simplest form of decision making process. Here every agent chooses its actions at the beginning of the game and the simultaneous execution of the chosen action by all agents.

A strategic game [45][46] consists of

- a set of players - in multi-agent scenario, the agents are assumed to be the players
- For each player, there is a set of actions
- For each player, the preferences over a set of action profiles

There is a payoff associated with each of the combination of action values for the participating players. The payoff function is assumed to be predefined and known in the case of a simple strategic game. It is also assumed that the actions of all agents are observable and is a common knowledge available to all agents. A solution to a specific game is the prediction of the outcome of the game making the assumption that all participating agents are rational.

The prisoner's dilemma is a best case for demonstrating the application of game theory in decision making involving multiple agents. The prisoner's dilemma problem can be states as

Two suspects involved in the same crime are interrogated independently. If both the prisoner's confess to the crime, each of them will end up spending three years in prison. If only one of the prisoner confesses to the crime, the confessor is free while the other person will spend four years in prison. If they both do not confess to the crime, each will spend a year in prison.

This scenario can be represented as a strategic game.

Players: Two suspects involved in the crime

Actions: Each agent's set of actions is {*Not confess, confess*}

Preferences: Ordering of the action profile for agent 1, from best to worst case scenario, is {*confess, Not confess*}, {*Not Confess, Not confess*}, {*Confess, Confess*} and {*Not confess, Confess*}. Similar ordering could be performed by agent 2.

A payoff matrix that represents the particular preferences of the agents needs to be created. Simple payoff matrix can be *u1{Confess, Not confess} =3, u1{Not confess, Not confess}=2. u1{Confess, Confess}=1, u1{Not confess, confess}=0.* Similarly the utility or payoff for agent 2 can be represented as *u2{Not confess, confess}=3, u2(Not confess, Not confess}=2, u2{confess, Not confess}=0* and *u2{confess, confess}=1.*

The reward or payoff received by each agent for choosing a specific joint action can be represented in a matrix format called as payoff matrix table. The problem depicts a scenario where the agents can gain if they cooperate with each other but there is also a possibility to be free if a confession is made. The particular problem can be represented as a payoff matrix as shown in Figure 10. In this case it can be seen that the solution "*Not confess*" is strictly dominated. By strictly dominated solution, it means that a specific action of an agent always increases the payoff of the agent irrespective of the other agents actions.

		Agent 2	
		Not Confess	*Confess*
Agent 1	*Not confess*	2,2	0,3
	Confess	3,0	1,1

Fig. 10. Payoff matrix in the Prisoner's Dilemma Problem

However, there can be variations to the prisoner's dilemma problem by introducing an altruistic preference while still calculating the payoff of the actions. Under this circumstance, there is no action strictly dominated by the other.

6.1 Nash Equilibrium

To obtain the best solution based on the constructed payoff matrix, the most common method employed is the Nash Equilibrium. Nash Equilibrium [47] can be stated as follows

A Nash Equilibrium is an action profile a with the property that no player i can do better by choosing an action different from a* of i, given that every other player adheres to a* of j.*

In the most idealistic conditions, where the components of the game are drawn randomly from a collection of populations or agents, a Nash equilibrium corresponds to a steady state value. In a strategic game, there always exists a Nash equilibrium but it is not necessarily a unique solution. Examining the payoff matrix in Figure. 11 shows that {*confess, confess*} is the Nash equilibrium for the particular problem. The action pair {*confess, confess*} is a Nash equilibrium because given that agent 2 chooses to confess, agent 1 is better off choosing confess than Non confess. By a similar argument with respect to agent 2 it can be concluded that {*confess, confess*} is a Nash Equilibrium. In particular, the incentive to have a free ride on confession eliminates any possibility of selecting mutually desirable outcome of the type {*Not Confess, Not Confess*}. If the payoff matrix could be modified to add value based on the trust or reward to create altruistic behaviour and feeling of indignation, then the subtle balance that exists shifts and the problem would have a multiple number of Nash equilibrium points as shown in Figure 11.

		Agent 2	
		Not Confess	*Confess*
Agent 1	*Not confess*	2,2	-2,-1
	Confess	-1,-2	1,1

Fig. 11. Modified Payoff matrix in the Prisoner's Dilemma Problem

In this particular case, there are no dominated solution and multiple Nash equilibrium would exist. To obtain a solution for the type of problem the coordination between the agents is an essential requirement.

6.2 The Iterated Elimination Method

The solution to the Prisoner's dilemma problem can also be obtained by using the iterated elimination method [48]. In this method, the strongly dominated actions are iteratively eliminated until no more actions are strictly dominated. The iterated elimination method assumes that all agents are rational and it would not choose a strictly dominated solution. This method is weaker than the Nash equilibrium as it finds the solution by means of a algorithm. Iterated elimination method fails when there are no strictly dominated actions available in the solution space. This limits the applicability of the method in multi-agent scenario where mostly weakly-dominated actions are encountered.

7 Coordination in Multi-Agent System

Coordination is the central issue in the design of multi-agent systems. Agents are seldom stand-alone systems and usually involve more than one agent working in parallel to achieve a common goal. When multiple agents are employed to achieve a goal, there is a necessity to coordinate or synchronize the actions to ensure the stability of the system. Coordination between agents increases the chances of attaining a optimal global solution. In [49] major reasons necessitating coordination between the agents were highlighted. The requirements are

- To prevent chaos and anarchy
- To meet global constraints
- To utilize distributed resources, expertise and information
- To prevent conflicts between agents
- To improve the overall efficiency of the system

Coordination can be achieved by applying constraints on the joint action choices of each agent or by utilizing the information collated from neighbouring agents. These are used to compute the equilibrium action point that could effectively enhance the utility of all the participating agents. Applying constraints on the joint actions requires an extensive knowledge of the application domain. This may not be readily available. It necessitates the selection of the proper action taken by each agent. It is based on the equilibrium action computed. However, the payoff matrix necessary to compute the utility value of all action choices might be difficult to determine. The dimension of the payoff matrix grows exponentially with the increasing the number of agents and the available action choices. This may create a bottleneck when computing the optimal solution.

The problem of this dimensional explosion can be solved by dividing the game into a number of sub-games that can be more effectively solved. A simple mechanism which can reduce the number of action choices is to apply constraints or assign roles to each agent. Once a specific role is assigned, the number of permitted action choices is reduced and are made more computationally feasible. This approach is of particular importance in a distributed coordination mechanism. However, in centralized coordination techniques this is not a major concern as it is possible to construct belief models for all agents. The payoff matrix can be computed centrally and provided to all of the agents as shared resource. The centralized coordination is adopted from the basic client/server model of coordination. Most of the centralized coordination techniques uses blackboards as a way in which to exchange information. Master agent schedules of all the connected agents are required to read and write information from and to the central information repository. Some of the commonly adopted client/server models are KASBAH[50] and MAGMA[51]. The model uses a global blackboard to achieve the required coordination. Disadvantage in using the centralized coordination is that of system disintegration resulting from a single point failure of the repository or of the mediating agent. Further use of the centralized coordination technique is contradictory to the basic assumption of DAI[52][49].

7.1 Coordination through Protocol

A classic coordination technique among agents in a distributed architecture is through the communication protocol. Protocol is usually in high level language which specifies the method of coordination between the agents and is a series of task and resource allocation methods. The most widely used protocol is the Contract Net Protocol [53] which facilitates the use of distributed control of cooperative task execution. The protocol specifies what information is to be communicated between the agents and the format of the information of dissemination is handled by the protocol. A low-level communication language such as KIF that can handle the communication streams is assumed to be available. The protocol engages in negotiation between the agents to arrive at an appropriate solution. The negotiation process must adhere to the following characteristics

1. Negotiation is a local process between agents and it involves no central control
2. Two way communication is available between all participating agents exists
3. Each agent makes its evaluation based on its own perception of the environment
4. The final agreement is made through a mutual selection of the action plan

Each agent assumes the role of *Manager* and *Contractor* as necessary. The manager essentially serves to break a larger problem into smaller sub-problems and finds contractors that can perform these functions effectively. A contractor can become a manager and decompose the sub-problem so as to reduce the computational cost and increase efficiency. The manager contracts with a contractor through a process of bidding. In the bidding process, the manager specifies the type of resource required and a description of the problem to be solved. Agents that are free or idle and have the resources required to perform the operation submits a bid indicating their capabilities. The manager agent then evaluates the received bids, chooses an appropriate contractor agent and awards the contract. In case of non-availability of any suitable contracting agent, the manager agent waits for a pre-specified period before rebroadcasting the contract to all agents. The contracting agent may negotiate with the manager agent seeking an access to a particular resource as a condition before accepting the contract.

The FIPA model [54] is the best example of an agent platform that utilizes the contract net protocol to achieve coordination in between the agents. FIPA - Foundation for Intelligent Physical Agents is a model developed to standardize agent technology. The FIPA has its own ACL (Agent Communication Language) that serves as the backbone for the high-level contract net protocol.

Disadvantage of the protocol based coordination is the assumption of the existence of an cooperative agent. The negotiation strategy is passive and does not involve any punitive measures which attempts to force an agent to adopt a specific strategy. Usually a common strategy is achieved through iterative communications where the negotiation parameters are modified progressively to achieve equilibrium. This makes the contract net protocol to be communication intensive.

7.2 Coordination via Graphs

Coordination graphs were introduced in [55] to serve as a framework to solve large scale distributed coordination problems. In coordination graphs, each problem is subdivided into smaller problems that are easier to solve. The main assumption with coordination graphs is that the payoffs can be expressed as a linear combination of the local payoffs of the sub-game. Based on this assumption, algorithm such as variable elimination method can compute the optimal joint actions by iteratively eliminating agents and creating new conditional functions that compute the maximal value the agent can achieve given the action of other agents on which it depends. The joint action choice is only known after the completion of the entire computation process, which scales with the increase in agents and available action choices and is of concern in time critical processes. An alternate method using max-plus which reduces the computation time required was used in [56]. This was to achieve coordination in multi-agent system when applied to urban traffic signal control.

7.3 Coordination through Belief Models

In scenarios where time is of critical importance, coordination through protocols fail to succeed when an agent with a specific resource to solve the sub-problem reject the bid. In such scenarios, agents with an internal belief model of the neighbouring agents could solve the problem. The internal belief model could be either evolved by observing the variation in the dynamics of the environment or developed based on heuristic knowledge and domain expertise. When the internal model is evolved, the agent has to be intelligent enough to differentiate between the change in environment due to other agent actions and natural variations occurring in the environment. In [20] [38], a heuristics based belief model has been employed to create coordination between agents and to effectively change the green time. In [57], evolutionary methods combined with neural networks have been employed to dynamically compute the level of cooperation required between the agents. This is based on the internal state model of the agents. The internal state model was updated using reinforcement learning methods.

A disadvantage using the coordination based on belief model for the agents is an incorrect model could cause chaos due to the actions selected.

8 Learning in a Multi-Agent System

The learning of an agent can be defined as building or modifying the belief structure based on the knowledge base, input information available and the consequences or actions needed to achieve the local goal [58]. Based on the above definition, agent learning can be classified into three types.

1. Active learning
2. Reactive learning
3. Learning based on consequence

In active and reactive learning, the update of the belief part of the agent is given preference over an optimal action selection strategy as a better belief model could increase the probability of the selection of an appropriate action.

8.1 Active Learning

Active learning can be described as a process of analysing the observations to create a belief or internal model of the corresponding situated agent's environment. The active learning process can be performed by using a deductive, inductive or probabilistic reasoning approach.

In the deductive learning approach, the agent draws a deductive inference to explain a particular instance or state-action sequence using its knowledge base. Since the result learned is implied or deduced from the original knowledge base which already exists, the information learnt by each agent is not a new but useful inference. The local goal of each agent could form a part of the knowledge base. In the deductive learning approach, the uncertainty or the inconsistency associated with the agent knowledge is usually disregarded. This makes it not suitable for real-time applications.

In an inductive learning approach, the agent learns from observations of state-action pair. These viewed as the instantiation of some underlying general rules or theories without the aid of a teacher or a reference model. Inductive learning is effective when the environment can be presented in terms of some generalized statements. Well known inductive learning approaches utilize the correlation between the observations and the final action space to create the internal state model of the agent. The functionality of inductive learning may be enhanced if the knowledge base is used as a supplement to infer the state model. The inductive learning approach suffers at the beginning of operation as statistically significant data pertaining to the agent may not be available.

The probabilistic learning approach is based on the assumption that the agent knowledge base or the belief model can be represented as probabilities of occurrence of events. The agent's observation of the environment is used to predict the internal state of the agent. One of the best examples of probabilistic learning is that of the Bayesian theorem. According to the Bayesian theorem, the posterior probability of an event can be determined by the prior probability of that event and the likelihood of its occurrence. The likelihood probability can be calculated based on observations of the samples collected from the environment and prior probability can be updated using the posterior probability calculated in the previous time step of the learning process. In a multi-agent scenario where the action of one agent influences the state of other agent, the application of using the probabilistic learning approach is difficult. This stems from the major knowledge requirement of the joint probability of actions and state space of different agents. With an increase in the number of agents, it is difficult in practice to calculate and infeasible computationally. The other limitation is the limited number of the sample observations available to estimate the correct trajectory.

8.2 Reactive Learning

The process of updating a belief without having the actual knowledge of what needs to be learnt or observed is called as Reactive Learning. This method is particularly useful when the underlying model of the agent or the environment is not known clearly and are designated as black box. Reactive learning can be seen in agents which utilize connectionist systems such as neural networks. Neural networks depend on the mechanism which maps the inputs to output data samples using inter-connected computational layers. Learning is done by the adjustment of the synaptic weights between the layers. In [59], reactive multi-agent based feed forward neural networks have been used and its application to the identification of non-linear dynamic system have been demonstrated. In [60] many other reactive learning methods such as accidental learning, go-with-the-flow, channel multiplexing and a shopping around approach have been discussed. Most of these methods are rarely employed in a real application environment as they depend on the application domain.

8.3 Learning Based on Consequences

Learning methods presented in the previous sections were concerned with understanding the environment based on the belief model update and analysis of patterns in sample observations. This section will deal with the learning methods based on the evaluation of the goodness of selected action. This may be performed by reinforcement learning methods.

Reinforcement learning is a way of programming the agents using reward and punishment scalar signals without specifying how the task is to be achieved. In reinforcement learning, the behaviour of the agent is learnt through trial and error interaction with the dynamic environment without an external teacher or supervisor that knows the right solution. Conventionally, reinforcement learning methods are used when the action space is small and discrete. Recent developments in reinforcement learning techniques have made it possible to use the methods in continuous and large state-action space scenarios too. Examples of applications using reinforcement learning techniques in reactive agents are given in [61] [62].

In reinforcement learning [63], the agent attempts to maximize the discounted scalar reward received from the environment over a finite period of time. To represent this, an agent is represented as a Markov Decision Process.

- A discrete number of states $s \in S$
- A discrete set of actions $a \in A$
- State transition probability $p(s' \mid s, a)$
- Reward function $R : SXA \to \infty$

The reward function can be written as $R = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$. The objective is to maximize this function for a given policy function. A policy is a mapping from the state to the action values. The optimal value of a specific state *s* can be defined as the maximum discounted future reward which is received by following a specific stationary policy and can be written as

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_t = \pi(s)\right] \tag{1}$$

The expectation operator averages the transition values. In a similar manner the Q value can be written as

$$Q^*(s,a) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t) \mid s_0 = s, a_0 = a\right] \tag{2}$$

The optimal policy can then be determined as argmax of the Q-value. To compute the optimal value function and the Q-value, the Bellman equation (3) and (4) is used. The solution to Bellman equation can be obtained by recursively computing the values using dynamic programming techniques. However, the transition probability values are difficult to obtain. Therefore the solution is obtained iteratively by using the temporal difference error between the value of successive iterations as shown in (5) and (6).

$$V^*(s) = \max_{\pi}\left[R(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s')\right] \tag{3}$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V(s') \tag{4}$$

$$V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right] \tag{5}$$

$$Q(s_t,a_t) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{6}$$

The solution to (6) is referred to as the q-learning method. The Q-value computed for each state action pair is stored in Q-map and is used to update the Q-values. Based on the Q-values, the appropriate actions are selected. The major disadvantage is that the exploration and exploitation trade-off must be determined. To build an efficient Q-map, it is essential to compute the Q-values corresponding to all the state-action pair. The convergence is guaranteed if all the state-action pairs have been visited infinite number of times (theoretical).

In single agent RL, the convergence and methodologies are well defined and proven. In a distributed MAS, the reinforcement learning method faces the problem of combinatorial explosion in the state action pairs. Another major concern is the information must be passed between the agents for effective learning. In [64], distributed value function based on RL has been described. A complete survey of reinforcement learning can be found in [65].

9 Conclusion

In this chapter, a brief survey of the existing architectures, communication requirements, coordination mechanism, decision making and learning in multi-agent systems have been presented. Rapid advances made in multi-agent system have created a scope for applying it to several applications that require distributed intelligent computing. The communication mechanism, coordination strategies and decision making are still in development stages and may offer a greater scope for further improvement and innovation. The increase in computing power has further increased the scope of MAS employability in real-world applications. These include many applications such as grid computing, robotic soccer, urban traffic signal control.

References

[1] Jennings, N.R., Sycara, K. and Woolridge, M, “A roadmap of agent research and development,” In Autonomous Agents and Multi-Agent Systems Journal, vol. 1, no.1, pp. 7-38, 1998

[2] Jain, L.C and Jain, R.K. (Editors), Hybrid Intelligent Engineering Systems, World Scientific Publishing Company, Singapore, 1997

[3] Mumford, C. And Jain, L.C.(Editors), Computational intelligence: Collaboration, Fusion and Emergence, Springer-Verlag, 2009

[4] Fulcher, J and Jain, L.C. (Editors), Computational Intelligence: A Compendium, Springer-Verlag, 2008

[5] Jain, L.C., Sato,M., Virvou, M., Tsihrintzis, G., Balas, V. And Abeynayake, C. (Editors), Computational Intelligence Paradigms: Volume 1 – Innovative Applications, Springer-Verlag, 2008

[6] Jain, L.C and De Wilde, P. (Editors), Practical Applications of Computational Intelligence Techniques, Kluwer Academic Publishers, USA, 2001

[7] Jain, L.C and Martin, N.M. (Editors), Fusion of Neural Networks, Fuzzy Logic and Evolutionary Computing and their Applications, CRC Press USA, 1999

[8] Tedorescu, H.N., Kandel, A. And Jain, L.C. (Editors), Fuzzy and Neuro-fuzzy Systems in Medicine, CRC Press USA, 1998

[9] Khosla, R., Ichalkaranje, N. And Jain, L.C. (Editors), Design of Intelligent Multi-Agent Systems, Springer-Verlag, Germany, 2005

[10] Jain, L.C., Chen, Z. And Ichalkaranje, N. (Editors), Intelligent Agents and Their Applications, Springer-Verlag, Germany, 2002

[11] Resconi, G. And Jain, L.C.,(Editors), Intelligent Agents: Theory and Applications, Springer-Verlag, Germany, 2004

[12] Nwana. H, “Software agents: An overview,” Knowledge and Engineering Review, vol. 11, no. 3, 1996

[13] Nikos Vlassis, “A Concise introduction to multiagent systems and distributed artifical intelligence,” Synthesis Lectures On Artificial Intelligence And Machine Learning, 1st edition, 2007

[14] P.Stone and M.Veloso, “Multiagent systems: A survey from the machine learning perspective,” Autonomous Robotics, vol. 8, no.3 , pp. 1-56, Jul 2000

[15] Ren,Z and Anumba, C.J, “ Learning in multi-agent systems: a case study of construction claim negotiation,” Advanced Engineering Informatics, vol. 16, no. 4, pp. 265-275, 2002

[16] Goldman, C.V , “Learning in multi-agent systems,” In Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, vol. 2, pp. 1363, 1996

[17] Eduardo Alonso, Mark D’Inverno, Daniel Kudenko, Michael Luck and Jason Noble, “Learning in multi-agent systems,” The Knowledge Engineering Review, vol.13, no. 3, pp. 277-284, 2001

[18] Bergenti, Federico and Ricci, Alessandro, “Three approaches to the coordination of multiagent systems,” In Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 367-373, Mar 2002

[19] Tien C.Hsia and Michael Soderstrand, “Development of a micro robot system for playing soccer games,” In Proceedings of the Micro-Robot World Cup Soccer Tournament, pp. 149-152, 1996

[20] Balaji P.G and D.Srinivasan, “Distributed multi-agent type-2 fuzzy architecture for urban traffic signal control,” In IEEE Internationa Conference on Fuzzy Systems, pp. 1624-1632, 2009

[21] Lynne E.Parker, “Heterogeneous multi-robot cooperation,” PhD Thesis, Massachusetts Institute of Technology, 1994

[22] Lynne E.Parker, “Life-long adaptation in hetergeneous multi-robot teams:response to continual variation in robot performance,” Autonomous Robots, vol. 8, no. 3, 2000

[23] Rafal Drezewski and Leszek Siwik, “Co-evolutionary multi-agent system with predator-prey mechanism for multi-objective optimization,” In Adaptive and Natural Computing Algorithms, LNCS, vol. 4431, pp. 67-76, 2007

[24] Damba. A and Watanabe. S, “Hierarchical control in a multiagent system,” International Journal of innovative computing, Information & Control, vol. 4, no. 2, pp. 3091-100, 2008

[25] Choy, M C, D Srinivasan and R L Cheu, "Neural Networks for Continuous Online Learning and Control," IEEE Transactions on Neural Networks, vol. 17, no. 6, pp. 1511-1531, 2006

[26] Balaji P.G, Sachdeva.G, D.Srinivasan and C.K.Tham, “Multi-agent system based urban traffic management,” In Proceedings of IEEE Congress on Evolutionary Computation, pp. 1740-1747, 2007

[27] A.Koestler, “The ghost in the machine,” Hutchinson Publication Group, London, 1967

[28] Paulo Leitao, Paul Valckenaers, and E. Adam, “Self-adaptation for robustness and cooperation in holonic multi-agent systems,” Trans. On large-Scale Data- &Knowl.-Cent. Syst. I, LNCS 5740, pp. 267-288, 2009

[29] O.Yadgar, S.Kraus and C.Oritz, “Scaling up distributed sensor networks: Cooperative large scale mobile agent organizations,” Distributed Sensor Networks: A Multiagent Perspective, pp. 185-218, 2003

[30] Michael Schillo and Klaus Fischer, “ A taxanomy of autonomy in multiagent organisation,” Autonomy 2003, LNAI 2969, pp. 68-82, 2004

[31] L. Bongearts, “ Integration of scheduling and control in holonic manufacturing systems,” PhD Thesis. Katholieke Universiteit Leuven, Belgium, 1998

[32] D.Srinivasan and M.C.Choy, “ Distributed problem solving using evolutionary learning in multi-agent systems,” Studies in Computational Intelligence , vol. 66, pp. 211-227, 2007

[33] Van De Vijsel, Michael and Anderson, John, “Coalition formation in multi-agent systems under real-world conditions,” AAAI Workshop – Technical Report, v WS-04-06, pp. 54-60, 2004

[34] Horling, Bryan and Lesser, Victor , “ A survey of multi-agent organizational paradigms,” Knowledge Engineering Review, vol. 19, no. 4, pp. 281-316, 2004

[35] Agogino, A.K and Tumer, K. , “Team formation in partially observable multi-agent systems,” NASA Ames Research Center, NTIS, 2004

[36] Budianto, “An overview and survey on multi agent system,” in Seminar Nasional “Soft Computing, Intelligent Systems and Information Technology” , SIIT 2005

[37] M.C.Choy, D.Srinivasan and R.L.Cheu, " Cooperative, hybrid agent architecture for real-time traffic signal control," IEEE Trans. On Systems, Man and Cybernetics-Part A: Systems and Humans, vol. 33, no. 5, pp. 597-607, 2003

[38] Balaji P.G, D.Srinivasan and C.K.Tham, " Coordination in distributed multi-agent system using type-2 fuzzy decision systems," in Proceedings of IEEE International Conference on Fuzzy Systems, pp. 2291-2298, 2008

[39] Susan E.Lander, "Issues in multiagent design systems," IEEE Expert, vol.12, no. 2 , pp. 18-26, 1997

[40] Robert A.Flores-Mendez, "Towards a standardization of multi-agent system framework," Crossroads, vol. 5, no. 4, pp. 18-24, 1999

[41] Yilmaz Cengeloglu, "A Framework for dynamic knowledge exchange among intelligent agents," AAAI Symposium, Control of the Physial World by Intelligent Agents, 1994

[42] Genesereth, M. and Fikes, R "Knowledge interchange format, Version 3.0 Reference manual," Technical report, Computer Science Department, Stanford University, USA., 1992

[43] Ginsberg, M "The Knowledge interchange format: The KIF of death," in AAAI Magazine, Fall 1991, vol. 12, no. 3, pp. 57-63

[44] Finin, T., Labrou, Y. and Mayfield, J "KQML as an agent communication language," in Software Agents, AAAI Press, pp. 291-316, 1997

[45] Gibbons.R, Game theory for Applied Economists, Princeton University Press, Princeton, NJ

[46] Greenwald. A, "The search for equilibrium in markov games," Synthesis Lectures on Artificial Intelligence and Machine Learning, 2007

[47] Nash, J.F, "Equilibrium points in n-person games," Proceedings of the National Academy of Sciences, vol. 36, pp. 48-49, 1950

[48] Gibbons. R, "An introduction to applicable game theory," The Journal of Economic Perspectives, vol. 11, no. 1, pp. 127-149, 1997

[49] H.S. Nwana, L.Lee and N.R. Jennings, "Co-ordination in multi-agent systems," Software Agents and Soft Computing Towards Enhancing Machine Intelligence, LNCS, vol. 1198, pp. 42-58, 1997

[50] Chavez, A and Maes, P. Kasbah "An agent marketplace for buying and selling goods," in Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'96

[51] Tsvetovatyy, M., Gini, M., Mobhasher, B and Wieckowski, Z "MAGMA: An agent-based virtual marketplace for electronic commerce," Applied Artificial Intelligence, vol. 11, no. 6, pp. 501-542, 1997

[52] Huhns M and Singh M.P, "CKBS-94 Tutorial: Distributed artificial intelligence for information systems,", DAKE Center, University of Keele, 1994

[53] R.G.Smith "The contract net protocol: High-level communication and control in a distributed problem solver," in IEEE Transactions on Computers, vol. 29, no.12, pp. 1104-1113, 1980

[54] P.D. O'Brien and R.C.Nicol, "FIPA-Towards a standard for software agents," BT Technology Journal, vol. 16, no. 3, pp. 51-59, 1998

[55] Guestrin C, Koller D and Parr R, "Multiagent planning with factored MDPs," Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2002

[56] L. Kuywe, S. Whiteson, B.Bakker and N. Vlassis, "Multiagent reinforcement learning for urban traffic control usin coordination graphs," ECML PKDD, LNAI 5211, pp. 656-671, 2008

[57] D.Srinivasan and M.C.Choy, “Neural networks for real-time traffic signal control,” IEEE Trans. Intelligent Transportation Systems, vol. 7, no. 3, pp. 261-272, 2006

[58] L. Lhotska, “Learning in multi-agent systems: Theoretical issues,” Computer Aided Systems Theory – EUROCAST’97, LNCS-1333, pp.394-405, 1997

[59] Gomez.F, Schmidhuber.J and Miikkulainen.R, “Efficient non-linear control through neuro evolution,” Proceedings of ECML 2006, pp. 654-662

[60] Jiming Jiu, Autonomous Agents and Multi-agent Systems, World Scientific Publication

[61] V.Vassiliades, A. Cleanthous and C. Christodoulou, “Multiagent reinforcement learning with spiking and non-spiking agents in the iterated prisoner’s dilemma,” LNCS 5768, pp. 737-746, 2009

[62] Gabel.T and Riedmiller. M, “On a successful application of multi-agent reinforcement learning to operations research benchmarks,” IEEE International Approximate Dynamic Programming and Reinforcement learning, pp. 68-75, 2007

[63] Sutton R.S and Barto A.G, Reinforcement Learning: An Introduction, MIT Press. Cambridge, MA

[64] Schneider.J, W.K. Wong, A.Moored and Riedmiller.M, “Distributed Value functions,” In the Proceedings of the sixteenth International Conference on Machine Learning, pp. 371-378, 1999

[65] Busoniu. L, Babuska.R and De Schutter. B, “ A comprehensive survey of multiagent reinforcement learning,” IEEE Trans. On Systems, Man and Cybernetics Part C: Applications and Reviews, vol. 38, no. 2, pp. 156-12, 2008