

Exploring Learning Patterns and Decision making Logic in a System Consisting of Multiple Autonomous Agents.

Raajas Sode , M.Sc -Computer Science , Mithibai College, Mumbai

Dr. Amol Joglekar , Guide- Mithibai College, Mumbai

Abstract

This article explores the general body of knowledge with respect to decision making in multi-agent systems. There are plenty applications of multi agent systems in the field of economics, video games, robotics, traffic control, disaster management, etc. The keen focus of this research would be on the applications in video games. The research work will also evolve the concept of objective spaces, a set of environment variables and function calls that are relevant to an agent's current goal. How an objective space changes when it is subjected to a dynamic environment throughout its life-cycle and helps it communicate and coordinate better with other agents. Finally, research will focus on conflict management and resolution of clashing interests of various agents and their objectives. Conflict management is discussed in order to make it clear that autonomous agents, no matter how precise and deterministic, will often face problems when they have to act collectively as a team, or have to deal with conflicting interests in the same environment.

Introduction

There are scientific studies that thoroughly explore the dissimilarities between decision making processes in autonomous software/hardware systems and humans. Although there are numerous ways in which humans and autonomous non-human agents behave similarly. Human beings boast about their sheer ingenious thinking and contemplative capabilities, share common characteristics in terms of deterministic decision making with autonomous agents.

A person introduced to a new environment will stumble frequently into un-perceived obstacles. This is purely due to a lack of prior information and habit for traversing in such an environment. All the decisions made here onward would be

based on involuntary reflexes, preconceived experiences and interpretations of past events and the interpretation of the current event. This behavior can easily be coded into an intelligent non-human agent, with certain limitations of course.

In a video gaming scenario, multiple agents are sent across the environment to search for and achieve certain utility. These are pre-programmed instructions and if-conditioned loops that reason out and act on perceptions that are made at the moment or based on prior percepts. Unlike humans, intelligent agents can seldom alter their goals and think consciously about their intention.

In every game, like a player, an intelligent NPC (Non-player character) can have certain state variables and function calls that can dictate the trajectory of the agent, without having to violate the rules of the game. To avoid a stochastic action sequence, entirely made by chance, the agent can make well informed decisions guided by certain protocols.

It is important also, for an agent to check his internal state from time to time, especially when interacting with another agent, to check whether there is any loss or damage done in course of the encounter, in order to classify the agent as hostile, and change trajectories in order to embrace a defensive state.

Episodically checking the major and minor checkpoints in a game raises the agent's expected utility and dictates what function it has to execute momentarily, as well as the consecutive functions it has to execute thereafter to achieve its main goal. This trajectory of course, can be changed and adapted to the current state of the environment, even if the agent does not have full visibility of the environment.

So ultimately, The Decision making process of any agent would be governed by only the following factors:

1. Its personal utility, a dynamic trajectory designed episodically according to the internal, external state as well as the intended utility and checkpoints of the agent.
2. State of the agents around (if any) and conclusions reached to after short communication bouts with other agents in order to collaborate or compete for a utility.

Limitations:

This paper will not be focusing on a goal oriented agent, as considering a multi-agent dynamic system with stochastic and non-deterministic components, it is assumed that most planning done prior would be pointless and sustain only momentarily as the environment state and the state of every such agent is affected by every other agent around it. Therefore, most of the decisions will be made with respect to the short term utility and long term utility of the agent in question, and also the utility of the agent in the perimeter defined around the agent, and as well as the changes in the environment.

A. Objective Spaces

(In this context, an objective space does not relate to what it means in other research material, and has to be considered entirely in the independent context that this paper represents.)

Objective spaces (in some studies referred to as Belief state) is the driving force of an intelligent agent. It is a list of objectives, protocols and instructions, function calls along with internal and environmental state variables that dictate to an agent its next actions and justifies its decided trajectory. It has several elements that would help an agent to plan its trajectory in a dynamic and unobserved environment. It is quite evident that a rational agent with information about an environment can have it easy, but as this paper focuses on just dynamic, non-deterministic, inaccessible environments, we will assume that there would be no use of any stored schemas or plans that could be used as a “best guess” for the agent to perform well.

B. Internal State Variables (ISV)

Every agent (especially in video games) has certain vital signs that are important to keep the agent active. These vital signs are measured and recorded in special variables called Internal State Variables. These variables together decide the fate of the agent. For example, in a First Person Shooter game, the vitals of an agent would be the health points of the agent, the armor integrity, movement speed, etc. These things, if compromised, could affect the entire existence of the agent itself.

Assume there is a First person shooter game like counter strike. In which case every agent, the player and NPC (non-player characters) have certain internal state variables like health points, moving speed, wealth and armor. If an agent decides to buy a Machine gun for the mission, which is usually heavier and makes an agent's movement slower, it has to weigh the odds of getting killed due to slow movement against the odds of successfully using the machine gun to getting maximum kills. As the machine gun affects the movement speed of the agent, the decision of buying the machine gun is up to the decision making process followed in the objective space, by weighing the outcomes, analyzing the perimeter (finding a good vantage point to take good shots) and such other factors.

Every decision taken by the agent to traverse in the environment and achieve maximum expected utility, should be taken in accordance to preservation of the internal state of the agent. The individualistic well being of an agent simply cannot be sacrificed in order to achieve

C. External State Variables (ESV)

External state variables contain the metrics of the environment, and other elements in the environment. Mostly A record of other agents that are present, obstacles, the hostile and friendly elements, and objects of use. It is here that every

piece of information about other agents in a multi-agent system is recorded.

External State Variables contain A subset called Dimensional Variables, Which contain dimensional factors of the environment like Time, Temperature and location, which is drawn out by considering that the start position of the agent is the origin.

In the FPS game discussed above, It is always known to every agent how many enemies are there (Competitive utility), how many team-mates does one have, (Collective Utility) and how to recognize between the two. However, most of the times, when an agents spots an enemy from a distance, there is no need to further probe (discussed ahead) the agent, as a hostile agent would always attack.

It is therefore important to receive sensory data when an agent perceives in the environment, record it, classify it, and segregate data points in accordance to the short term and long term objectives.

Even though this paper doesn't focus on planning due to the assumption of a dynamically changing environment, it is possible to include an abstract mapping element as one of the environment state variables, for storing a map of the environment, which will be subjected to changes, but this map too, would be incomplete, because when the agent does move about in the environment, it is not quite certain that the agent would have enough resources (in case it is a big environment) or have an intention to traverse the whole environment in order to create a solid mapping of all locations and elements.

Objective Sequence

Analogous to the percept sequence which records percepts from sensory input, Objective sequence maps agents and objects that are important and to be dealt with as top priority, in order to achieve expected utility assigned to the agent.

D. Perimeter Check Function

The first thing an agent should do in an environment is a perimeter check. (The larger the perimeter, the more information the agent must process). A perimeter check should be a short scan in order to perceive all the objects that are present in its field of vision and identify and classify them to prove "useful". The list of objects made is then segregated within the objective space where agents, objects and obstacles are categorized and their use is analyzed as per the itinerary of the agent presented in the objective space.

The perimeter for an agent cannot be too wide as it might be an overwhelming amount of information to process. If the dynamic environment is large enough, the perimeter scan can take place in bouts as the agent moves around, updating the object list as it moves about, eliminating those that are not a part of the trajectory and preserving those that would prove to be useful to create the action profile.

A perimeter check could prove to be an important function as it would allow an agent to classify hostile, friendly and neutral elements when travelling in a previously unexplored environment and create a trajectory to move forward in order to achieve its utility.

E. Probe Function

The probe function is where an agent tries to approach a listed agent/object in order to further create and clarify its trajectory in order to achieve its expected utility. Probing could be just initiating communication with another agent (to exchange information and update collective and individual utilities), manipulating an object that seems to be useful, or even avoiding an obstacle in some cases. Probing is simply trying to retrieve more information about any element in the environment within the limitations of an agent's perception capabilities. It is the first initial **check** done by the

agent before any further engagement with an unfamiliar element in its perimeter.

F. Dynamic Planning Module

After probing comes the decision making process. Here an agent, after gaining all the information about a perceived object, would create a small process on the fly in order to quickly execute and gain the expected utility.

Like the Architecture of the InterRRap [3], where there is a layer for local planning, which uses sensory percepts of the environment to create a plan from one of the stored schemas available to implement a plan in order to achieve an objective in the given environment, It is quite evident that a planning layer in an agent architecture should be present to guide the agent from a “big picture” perspective.

When agents are subjected to a dynamically changing environment, where most of the environment is inaccessible, it would be quite unwise to pre plan every trajectory the agent would take in advance based on the current percept queue as other agents and stochastic elements would tend to re-orient or re-position themselves, rendering the plan pointless to execute further.

Although it is a valid option to store general schemas to pre-plan entire trajectories in order to reduce the effort of dynamically creating a plan drastically, static planning might prove to be a limitation as the schemas stored would be limited and of a stringent variation, which couldn't capture all possible scenarios a dynamically changing environment has to offer.

If we construct a Dynamic planning module which breaks down larger objectives into smaller goals and creates plans episodically for those smaller goals in a way that makes it quick for an agent to execute, gain closure on the smaller goal, and move on to the next miniature goal, while discarding the previous plan or leaving it open to manipulation for the next task.

G. Body Of Knowledge

This is the memory intensive module where actions, preferences and facts about the current environment are currently stored. This might seem like it's just a block of memory, but it would constantly change under the pretext that each life cycle of the agent would bring some new information in light. To create “experience”. It could have a series of actions prepared after each successful trajectory traversal, or have the record of an interaction with another agent, in order to follow the same protocol when approaching the agent again, or perhaps approaching the agent differently. It is a body of Knowledge where the agent stores most of what it has learned.

Anything and everything cannot simply be added to an agent's body of knowledge. What is added should be purely restricted to what the agent's intentions are, and what is useful for the agent considering the kind of tasks it keeps undertaking.

For the nitty-gritty decision making, it is proposed often to use a decision network, with the help of stored utility processes from the past in order to achieve maximum expected utility. However, in an unknown environment, it is often considered wise to consult other surrounding agent and retrieve as much information from them in order to maximize performance. It performs the function of a tree, but it does it so for iterative operations [citation].

Conflict Management

Autonomous Agents in an environment that is dynamic and inaccessible, demands that there is some sort of communication among different agents. Unless there is a very well laid out plan for each agent's itinerary and all assorted variables and resources are assigned in order to avoid all

conflict, there are still chances that there is some form of friction between agents or even a need to cooperate in case there is some sort of problem that two or more agents could put their heads together and achieve.

A. Cooperative Scenario

In any team sport or video game, Where there are multiple players or Multiple Non-Player Characters, there is to be some kind of Competitive Utility to be achieved, an agent needs to understand its instantaneous short-comings and needs to take action in order to delegate a certain task and resources to a more capable agent. In the same way, Offer assistance to those agents around that aren't quite able to make their way through. Ideally, where every task is equally divided, and there is no chance of interaction between agents, it would seem simple enough to execute and achieve a competitive utility, assuming nothing goes wrong. As humans cannot survive in complete isolation, there is no scenario that exists in any multi agent system which is dynamic, inaccessible and non-deterministic enough to not require any cooperation or interaction between agents.

B. Competitive Scenario

A competitive scenario is where all agents have to compete for resources in order to maximize their own utility before other agents. This requires either a brute force approach where there complete domination by one of the agents, compromising the integrity of others, or there is simply a random banter where the outcome would be entirely left to the stochastic nature of the competition.

C. Recognizing Non-Intentional or “hollow” conflict

Before delving into Conflict management, there is something that needs to be addressed in order to avoid unnecessary processing or preserving a state of conflict.

There are sometimes in a video game where there is an unnecessary state of conflict initiated, by perhaps friendly fires in a First-Person-Shooter game, or a triggered situation where there is a banter for no reason. In a situation like this there is a loss of resources, vital information, and processing time that could be used elsewhere to accomplish individual and collective utilities. Therefore, it is important for every agent to recognize and perform constant “checks” as to avoid any conflict before there comes a time to neutralize one. These checks involve checking one's priorities, utilities, and trying to proactively engage other agents into a collaborative effort if possible in order to achieve one's own individual utility, and helping other agents to perform the same.

It is also important to make sure that there is no conflict for no reason at all. For example, an agent might trigger a friendly fire or a grenade without the intention of harming his own team-mate. That does not mean that the agent is hostile. It is simply an error of judgment or a miss-fire that should be recognized through keeping a constant communication grid alive and preserve harmony among agents.

In a scenario in an FPS game, there are several instances where there are bugs in a game or there are friendly fires that tend to harm the vitals of a player or an agent. It is evident that sometimes in an inaccessible or unobservable part of an environment an agent tends to act irrationally and fire in the direction from where the act of unintentional hostility was committed. In this scenario, an agent could clearly take cover, perform a check on its vitals, and then establish a communication in order to ensure that the presence of a friend and not any hostile agent.

Recognizing unintentional conflict by performing constant episodic checks and keeping a robust communication system alive can be the key to allowing agents to perform at their highest without having to engage in conflict in the first place.

D. Conflict Resolution.

In a multi-agent system, agents are required to achieve a certain collective utility in order to accomplish certain goals during their life cycle. It is evident that considering the magnitude of the task is huge enough for all the agents to handle, the agents would have to coordinate and cooperate with one another in order to achieve their collective goal. In such a case it is inevitable that in a dynamic environment with rational agents who are self-aware of their core competencies and capabilities, there will be certain amount of friction in situations that have a conflict of interests between the agents.

In such a situation where there is a conflict among agents, there is a need for a rather balanced solution, rather than just a brute force approach which compromises the structural or functional integrity of one or more other agents, and preserve the harmony among the multi-agent system.

When agents communicate, in a real or simulated world, the only conflict that can arise can be regarding resources (physical) or regarding ideas (knowledge/opinion)[6] where agents are assigned to conflicting goals or agents themselves have conflicting opinions while attacking one single goal depending on their past experiences.

In any case of conflict there are six strategies proposed by Tessier [6] in order to handle such conflicts:

1. Flight: fleeing one of the two opponents
2. Destruction: Takeover one of opponents
3. Subservience: submission (giving up) by one of the opponents
4. Delegation: Delegate the judgment task to a third party
5. Compromising: obtains result by both agents approaching a middle-ground.
6. Consensus: obtains the agreement of opponents

In such a situation, it is easy to act irrationally, as both agents would want to have it their way. Most studies suggest the above six strategies in order to resolve conflict. Although [5] Works presented by Liu and Colleagues in 1999 suggests that the classification or identification of the type and intensity

of conflict makes conflict management very easy and efficient, as it narrows down the search to an even smaller number of options to choose from. The second factor that must be considered is the autonomy level of agent. The third factor being the agent's preferences. Their work proposes that all these factors when measured would initially help in focusing on the relevant strategy that could be used to resolve the conflict.

One of these paper have been written by Tang [Tang, Alicia. (2013). *A Framework for Conflict Resolution in Multi-Agent Systems. Computational Collective Intelligence. Technologies and Applications*. 8083. 195-204.] And colleagues Which summarizes most strategies used in the past and one such strategy is proposed in which specifies an algorithm that can allow an agent to decide what decision to make when conflict arises.

As a model proposed in the above paper defined two factors based on which all other conflict resolution strategies would be decided. The proposed framework decided two metrics on the basis of which the conflict resolution strategies are decided.

1. Confidence Level of a certain opinion which varies from one agent to another depending on pre-defined confidence factors.
2. Intensity of The Conflict itself: Which defines the magnitude level of the conflict between the two agent's opinions.

This paper defines two levels of conflict strength,

1. Strong Conflict: when two agents conflict more than 50% of their decisions or opinions. Under this type of conflict we have two situations:
 - a. A conflict between two agents with both having either high level confidence or both having low level confidence. In such a situation, the conflict resolution responsibility falls upon a higher authority or delegate. This strategy is called **Delegation**.
 - b. Conflicting agents with different levels of confidence. The proposed strategy in this case would be **Forcing**

2. Weak Conflict: When two agents conflict more than 50% of their decisions or opinions. Under this type of conflict there are three situations:
 - a. Conflicting agents with High Level of Confidence (HLC). Proposed strategy here is **Negotiations**.
 - b. Conflicting agents with High Level Confidence and Low Level Confidence (LLC). The suitable strategy is **submitting**.
 - c. Conflicting agents with Low Level Confidence in both agents. Here the strategy is **ignoring**.

Further the paper defines an algorithm that makes it easy for an agent to decide upon a certain conflict resolution strategy in order to obtain the best outcome. This is done by using nested if conditions to test the conflict strength first, later within which would be decided the confidence levels of both the agents.

Exploring Conflict Resolution in a Utility-Centric Scenario.

When we talk about Conflict resolution in intelligent systems, it is simple enough to narrow down the situation to a brute force algorithm that acts in a biased manner considering two strict metrics, ignoring the fact that the approach might not work in the favor of the utility.

There are certain factors that need to be considered while applying strategies that are specified by Tang [citation] .

1. Intelligent agents that learn from experience and reinforcement, when subjected to a stringent policy or protocol of conflict resolution, might not perform well in the future as they would be trained to submit/dominate/ignore/negotiate, in a situa-

tion that might not be the same as the previous one.

2. Algorithms proposed would be highly inefficient if the conflict resolution happens among team-mates that have to coordinate and cooperate in the future to achieve the collective utility of the team rather than just a certain individual goal. Agent's in this scenario would be less willing to cooperate in the future in order to avoid potential conflict, as a consequence to the completely dominative/submissive result of the previous conflict.

Therefore, considering the harmony that is to be preserved in a coordinating and cooperative multi agent system, this paper proposes a different perspective from which a Multi Agent System can make better conflict resolution decisions.

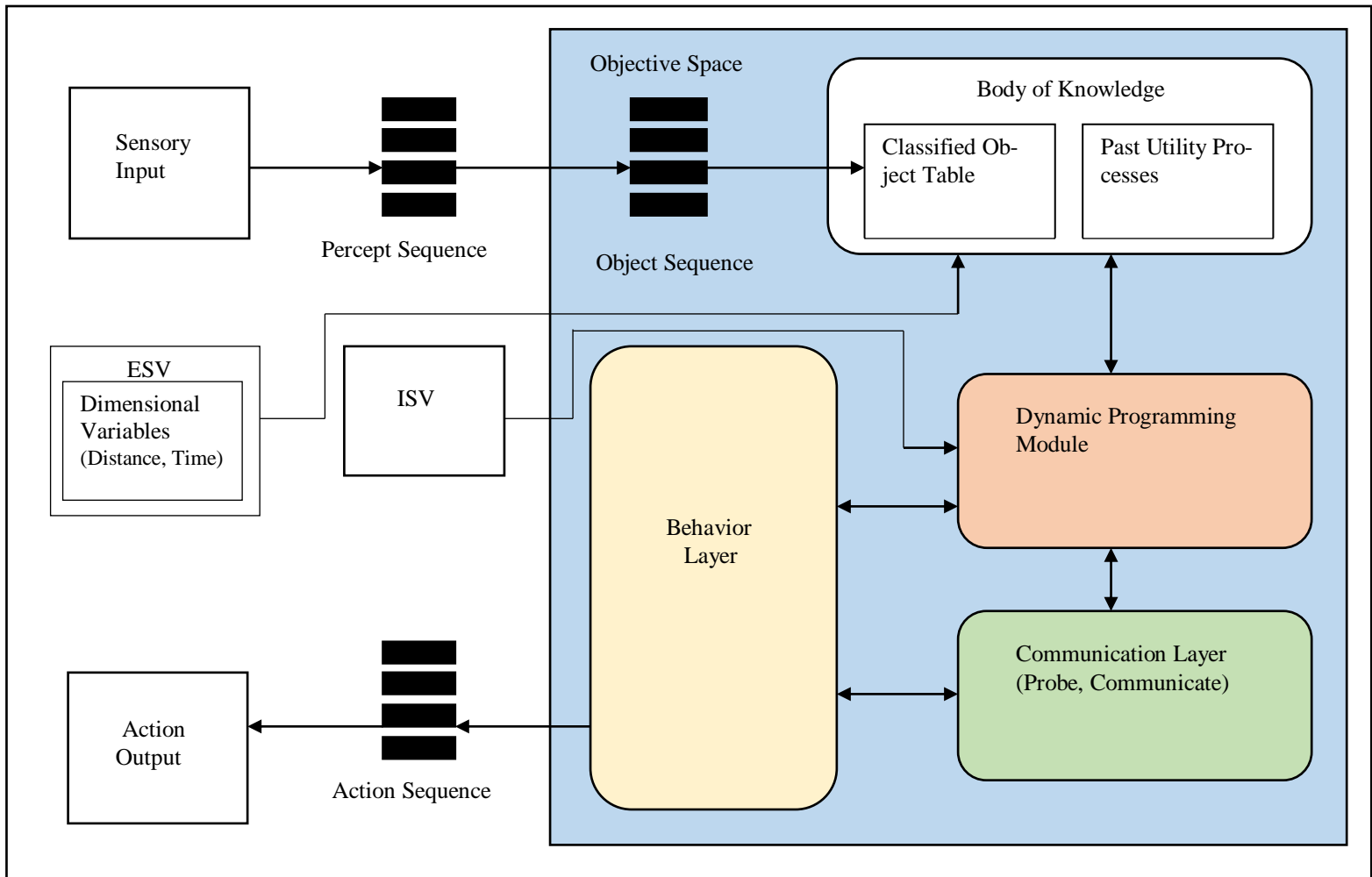
Two metrics are proposed here what should be used when judging a situation of conflict or resources or opinions:

1. **Adherence to Utility:** The need of the hour. All agents need to understand the conflict and make the decision that in the end stands in coherence with the utility trajectory that has been planned. It simply does not matter what is the confidence level or the level of conflict, but if an agent's decision (no matter how experienced) aligns itself with the collective utility, should prevail.
2. **Priority:** Each task assigned to each agent in a multi-agent system where collective utility is concerned, is assigned with a certain priority score. The priority score decides what task has to be completed first, or holds the highest contribution to the collective utility. Again, here too, the more intensive measure to be taken should be in coherence of the utility that has to be accomplished as a cooperative and coordinated team, rather than an individual entity. Algorithms designed keeping in mind the utility and the priority would perform much better in a scenario of coordination.

Every agent must be equipped with enough capability to take fair judgment and act rationally, in a situation of conflict, avoiding delegation. There should be routines built in the communication module of the agent (proposed above), that checks the internal state of every other agent, the external state of the environment, where they are in the goal accomplishment timeline and what is the priority of each agent and who needs the resource (over which the conflict has taken place) in order to maximize collective utility. This can be achieved by having a peer-to-peer model of communication between agents in case of conflict, where each agent is empathetic about the utility and its fellow agents rather than its own individual goals. This of course, assuming that the multi-agent system is not a competitive one.

Proposed Architecture

- The **Dynamic Planning module** breaks down each piece of information in order to re-assemble routines and action calls to complete miniature tasks to and gradually achieve its maximum utility. Everything the Agent does is decided by the Dynamic Planning module. No process or communication happens outside of the Dynamic Planning Module. As soon as the module has gained closure on one task and has a lead to the next task, the current routine is either discarded, or stored in the body of knowledge, depending on its utility score.
- The **Communication Layer** Contains protocols that aid the agent in communicating with other agents and the environment. The **Probe Function**, described earlier, resides in this layer. It is here that all the information passed between the agent and the environment are created, sent and received. The Communication Layer Sends messages to communicate with other agents.
- The **Behavior layer**, unlike in other architectures, is the forefront of the architecture. It is the interface of the Agent and it is the place where most instructions and function calls of the agent reside. Sending messages in the form of signals, implementing different probe functions for different objects, and implementing subroutines as decided by the Dynamic Planning Module. The Behavior layer sequences each action before they are sent to the action sequence
- The Sensory data taken in from the preceptors are passed to a sequence, called the percept sequence, which hold all the percepts.
- All objects from the percept sequence (agents, obstacles, etc.) are passed on to a sequence inside the objective space called the object sequence, while all dimensional percepts like location, temperature, time, etc. is passed on to the ESV (External State Variables).
- The **Object Sequence** that holds all objects of concern passes them on to a table called the **Classified Object Table**, inside the body of knowledge, which segregates the objects into agents, obstacles and objects concerned with the utility. This makes it easy to passed named objects on to the dynamic planning module.
- Facts are either discarded, or stored in the body of knowledge, depending on its utility score.
- The Body of knowledge provides all insights received from the percept sequence and from past experiences to the Dynamic Planning Module.



- The Dynamic Planning Module is the center for all calculations. Which is why it is very important when every plan is created, to consider the internal state variables, making sure that the **vitals** of an agent are not compromised in any way.

Although most components in the above architecture don't act like layers, but rather as modules, as they are independent of each other function-wise, and only co-dependent for information.

(Each Idea presented in the Diagram is a culmination of multiple ideas and other research papers, which are listed in the reference section of this paper.)

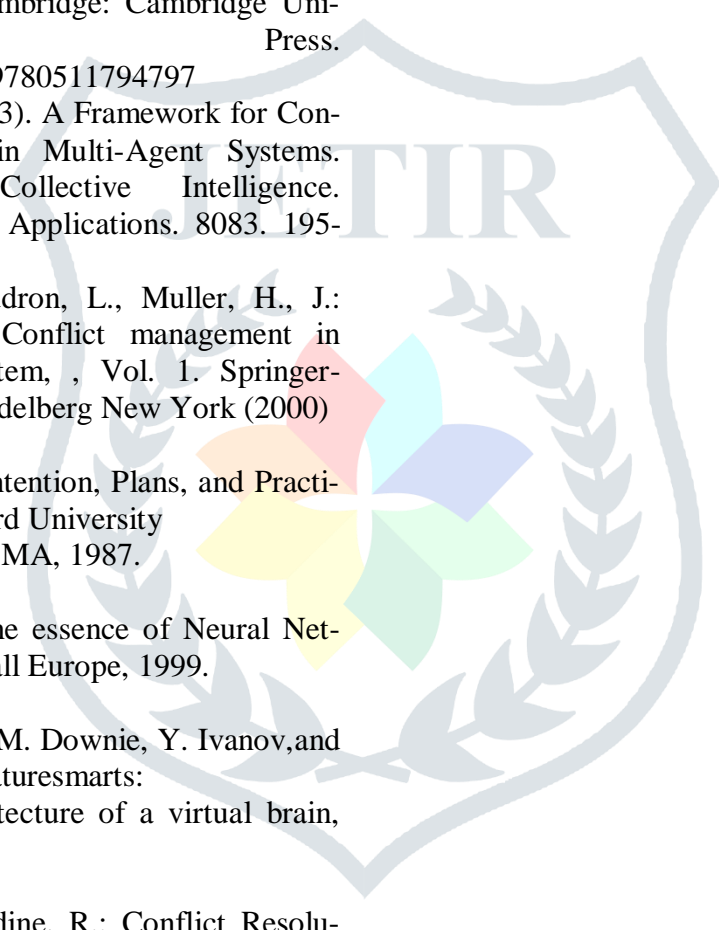
every action of the agent is driven from that point onward.

While building Multi-Agent Systems, it is to be kept in mind that working in complete harmony is the only scenario which agents should strive to work towards individually as well as collectively. Especially in a case of collective utility, as any loss of resource or the loss of an agent in the midst of a heavy competition can cost the operation time and its utility. This involves checking priorities and maximized utilities of each agent in the case of a conflict. It is apparent that not every agent can get its way, but collectively, a solution can be chalked out logically and rationally that proves to be advantageous to all the agents in the long term.

In future works this research will lay a keen stress on possible implementations of the architecture proposed along with improvements.

References

of the 16th European Conference on Artificial Intelligence (ECAI) (2004)

- 
- [1] Russell, S. J., Norvig, P., & Davis, E. (2010). Artificial intelligence: a modern approach. 3rd ed. Upper Saddle River, NJ: Prentice Hall.
- [2] Kaufman, Maïke “ Local Decision-Making in Multi-Agent Systems” Balliol College , University of Oxford, 2010.
- [3] Loland.S.Karl, Intelligent Agents in Computer Games Norwegian University of Science and Technology, June 2008.
- [4] Poole, D., & Mackworth, A. (2010). Artificial Intelligence: Foundations of Computational Agents. Cambridge: Cambridge University Press.
doi:10.1017/CBO9780511794797
- [5] Tang, Alicia. (2013). A Framework for Conflict Resolution in Multi-Agent Systems. Computational Collective Intelligence. Technologies and Applications. 8083. 195-204.
- [6] Tessier, C., Chaudron, L., Muller, H., J.: Conflict agents, Conflict management in Multi Agent System, , Vol. 1. Springer-Verlag, Berlin Heidelberg New York (2000)
- [7] M. E. Bratman. Intention, Plans, and Practical Reason. Harvard University Press, Cambridge, MA, 1987.
- [8] Robert Callan. The essence of Neural Networks. Practice Hall Europe, 1999.
- [9] R. Burke, D. Isla, M. Downie, Y. Ivanov, and B. Blumberg. Creaturesmarts: The art and architecture of a virtual brain, 2001.
- [10] Crawford, D., Bodine, R.: Conflict Resolution Education A Guide to Implementing Programs in Schools. Youth-Serving Organizations, and Community and Juvenile Justice Settings Program Report (1996)
- [11] Wagner, T., Shapiro, J., Xuan, P., Lesser, V.: Multi-Level Conflict in Multi-Agent Systems. Lecture Notes in Artificial intelligent, Vol. 4386. Springer-Verlag, Berlin Heidelberg Germany (2007)
- [12] Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An Integrated Trust and Reputation Model for Open Multi-agent System. In: Proceeding