Performance Measurement of Multi-Agent Systems

Muhammad Mohiuddin
University of Passau
Passau, Germany
mohiud01@gw.uni-passau.de

Abstract - A multi-agent system can greatly increase performance and reliability of a system due to several reasons like distributed nature, responsiveness to environment, and the ability for reuse. These characteristics are associated with multi-agent systems due to their flexible and intelligent nature. All these capabilities do not come without challenges. One of the biggest challenges that arises due to the dynamic behavior of multi-agent systems is the difficulty to quantify their reliability and dependability, or in other words performance.

This article discusses agents and multi-agent systems, their classification according design parameters, multiple methods of performance quantization, and factors which affect them.

Keywords: Software Agent, Multi-agent Systems, Agent architecture, Performance Analysis, Performance Parameters

1. INTRODUCTION

A. Agent

Researchers in the field of artificial intelligence have so far failed to agree on a consensus definition of the word "Agent" [1]. Two of the most regarded definitions are presented from Russel [2] and FIPA [3]. Russel and Norvig have defined and agent as an entity, which could perceive the environment utilizing the sensors it can access. These agents can then make decisions autonomously or in conjunction with other agents present and can affect that environment by using actuators.

FIPA defined agent as an agent is the fundamental actor in a domain. It combines one or more service capabilities into a unified and integrated execution model which can include access to external software, human users and communication facilities. These two definitions although agreed and highly regarded, still have not been able to develop a consensus. 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 [4]. These definitions provided do 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 using sensors and the resultant actions of the actuators. Environment in which an agent is present is an integral part of its design. All the inputs are received directly as a consequence of the agent's 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 attributes protect the internal states of agent from external influence. It isolates the agent from instability caused by external disturbances [5,6].

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 [5,6].

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 [5,6].

Pro-activeness: Agent must exhibit a good response to opportunistic behavior. This is 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 [5,6].

Social behavior: 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 are 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 [5,6].

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

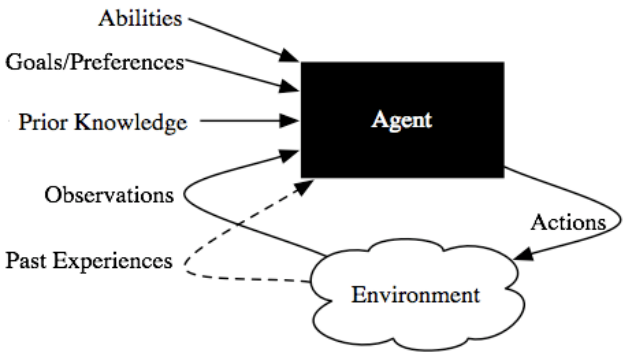

Figure 1 Agent and its interaction with environment (Wikimedia)

B. *Distributed Artificial Intelligence*

Distributed or Decentralized Artificial Intelligence (DAI) is a sub field of Artificial Intelligence. It is a predecessor to the multi-agent systems. It has gained importance recently, due to its ability to solve complex real-world problems. Along with multi-agent systems, it has two other branches, parallel AI and distributed solving. Multi-agent system is different from the other two, due to the involvement of agents (autonomous strategies and interconnection). Meanwhile, parallel AI is concerned about speed of operation and computational efficiency of an application. And distributed problem-solving focuses on predefined strategies and connection schemes to achieve decentralization [7][24].

C. *Multi-Agent System*

A Multi-Agent System (MAS) is an extension of the agent technology where a group of loosely connected autonomous agents act in an environment. These agents can either have a common goal, or they can have their local goals. Depending on the scope of goal, they may cooperate or compete, and decide whether to share or not share their local knowledge with each other. Whatever the case, there is always a coordination mechanism working in the system between agents.
Multi-agent systems have been widely adopted in many application domains because of the offered advantages. Some of the benefits available by using MAS technology in large systems [7,8] 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 agents 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 neighbors. This necessitates that each agent must predict the action of the other agents to decide the optimal action that would be goal directed. This type of concurrent learning could result in non-stable behavior 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 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 consist 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 f or 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.

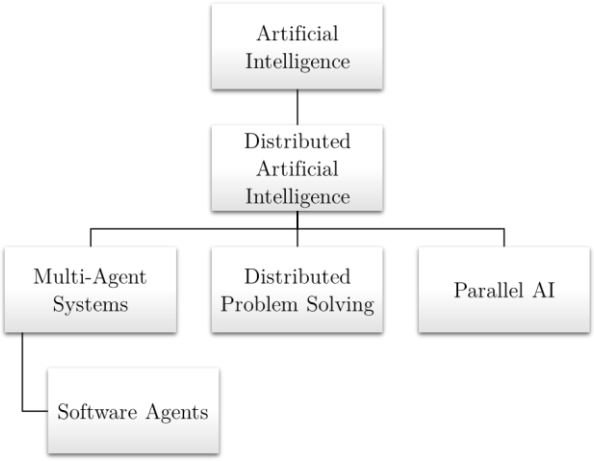

Figure 2 Classification of MAS inside AI

2. Design of Multi-agent Systems

MAS can be classified based on different design parameters [9-13]. Here we will discuss only two.

Multi-agent systems can be heterogeneous or homogeneous, depending of the type of agents involved. If all agents in a system are similar, in terms of their capabilities and functionalities (or perform similar tasks), then a system is called homogenous system [14]. Otherwise, a system whose agents are different in terms of capabilities and functionalities is called a heterogenous system [15].

Secondly, multi-agent systems can be classified based on their communication architectures as mesh or hierarchical. In a mesh, all agents can communicate with all other agents in a system. While, hierarchical system is more layered. And an agent can only connect with some certain agents (usually a parent and children).

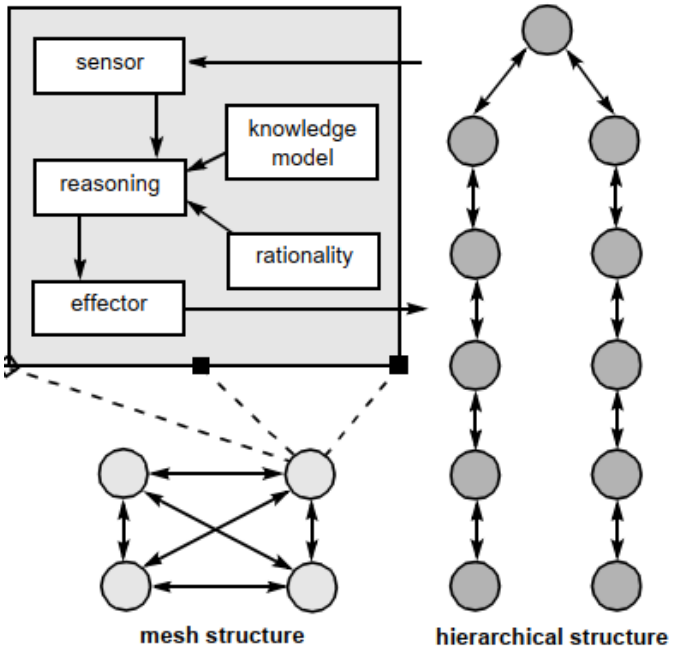

Figure 3 Examples of multi-agent coordination [24]

3. Performance

A. Performance Indicators

1. Computational Efficiency: Amount of time and resources an agent requires to perform all its assigned tasks.
2. Coordination Mechanism: Organizational structure in which agents have organized themselves. This also includes computational and communication costs involved.
3. Rationality model: Capability of optimization and evaluation techniques an agent employs.
4. Knowledge model: Data organization and manipulation capabilities of an agent.

These parameters which are used to quantify the above-mentioned indicators are discussed in the next section.

B. Performance Parameters

In this section a number of parameters associated with the performance of a MAS are mentioned. These parameters are directly or indirectly associated with the performance of a MAS. These parameters are identified from the general architecture and implementation of a MAS [16, 17].

A. *Number of Agents:* Number of agents in a MAS directly affects its performance. This factor is also directly linked with the agents' communication mechanism and the overall complexity of a MAS. If the number of agents is too high, then the complexity of a system increases, and it requires additional effort to maintain the same performance levels.

B. *Optimizations:* Agents employ several optimization techniques. These optimizations can be local, but still affect the performance of other agents and the system, depending on the level of coordination and its mechanism.

C. *Active Time of Agent:* When an agent is executing the logic of its assigned responsibilities, it is said to be in Active State. The amount of time an agent spends in this state is a good parameter to calculate its performance. This time during which an agent is active is also called computational time.

D. *Active Memory of Agent:* The amount of memory an agent requires to fulfill its assigned responsibilities is also a good measure of an agent's performance. Active memory is the sum of all memory allocations an agent has made for objects utilized during the active state.

E. *Status of Agent:* An agent can be in any of several states during the lifetime of its operation i.e. Active state, Wait state etc. The number of times

an agent shifts from one state to another, can have an effect on its performance. Poor performance will be delivered if an agent requires too many state shifts, to carry out a task.

F. *Coordination between Agents:* Coordination between agents means, how efficiently the agents, which are part of a multi-agent system, can break down a global assigned task into smaller local tasks among themselves. This also includes how the agents communicate among themselves, while the local tasks are being executed. Influence of this parameter on performance increases even further, in a homogenous system. As, all agents have same capabilities and functionalities, and division of responsibilities becomes a bigger challenge [18].

G. *Communication Mechanism:* Multi-agent system's performance is affected by the communication mechanisms it employs. Communication mechanism includes protocols, techniques and schemes used to enable communication not only between the master and agents, but also between agents. Length of messages, overhead or number of physical machines involved can all have an effect.

H. *System Management Mechanism:* In a multi-agent system, a global state must be maintained. This global state is used to keep record of individual agent's states, roles, progress and status of task execution. Also, some global properties like work completed, work remaining, progress of work through time and the percentage of active agents in the entire system. All these tasks can seriously affect the performance of a system, depending on the amount of overhead involved in these management tasks.

I. *Reliance and Subordination between Agents:* In a MAS not everything is parallelizable, and all agents are not completely independent and autonomous. The non-parallel tasks (or reliance of one agent on another agent's output) can affect the performance of a system in a critical manner. Mostly, these bottlenecks arise due to the involvement of global state even during the execution of a parallelized task. For example, in this scenario, one or several agents must shift to Wait state, until another weaker agent has completed a segment of their task. These dependencies can be analyzed during runtime, using a management system, or during design time using static analysis [19].

4. Performance Testing

Performance testing means to guarantee correctness and throughput of a MAS, for all environments and workloads it is designed for.

A. Techniques

Testing techniques used in the field of software engineering in general can also be applied to test and guarantee the validity and correctness of a MAS. These testing techniques which can be used in the domain of MAS are listed below [20-23]:

1. Theorem proving
2. Model checking
3. Static Analysis
4. Testing
5. Runtime Monitoring

Out of the above 5 techniques, the one which is most important for MAS is runtime monitoring.

Theorem proving: It is a formal technique to guarantee validity of a state or states under all defined conditions. For this purpose, specialized logic must be designed for all states. This is a rather difficult approach, to the dynamic and open environment nature of MAS. This approach is rarely used except during the development of safety critical systems.

Model checking: It is also a formal technique used during the development of MAS. These tests guarantee that a tested property holds in all possible states. In other words, all states and state transitions of a system are modeled. The coverage of these tests depend on how much system model has been properly defined. Complete test coverage is guaranteed for completely modeled system. Although, completely modeling a MAS is practically very difficult. Primarily, due to its flexible behavior and open environment nature. For model checking a language called MABLE can be used, which allows to generate Promela models. These Promela models can then be checked using SPIN.

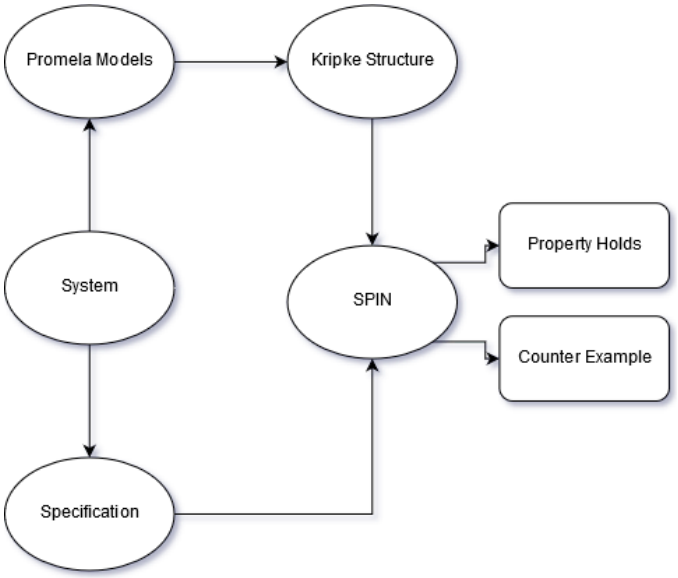

Figure 4 Basic flow of Model Checking

Static Analysis: It is used to analyze the source code of the MAS without executing it. The main target of this testing technique is the data and logic flow of the system. Static analysis has a very low coverage, because of the distributed and autonomous nature of MAS. Despite of low coverage, it is still used extensively, because it is a highly automated process, and often integrated right into the compilers.

Testing: Standard software testing techniques are also used in the domain of MAS. These include unit testing, integration testing and instrumentation testing.

Runtime monitoring: It is a popular technique used for testing MAS. This technique involves several approaches to monitor and evaluate the behavior of a MAS at runtime. The main reason of its popularity is its flexibility and ease of use, due to its compatibility with simulated environments or simulation software.

- One of the approaches is the use of conditional statements or asserts inside the source code. When these conditional statements are executed, they determine whether the output of the program is valid. If condition holds true then the normal execution of the program can continue, otherwise the program enters a recovery mode, or simply crashes to indicate a failure. The problem of this approach is it is highly local, it's difficult to validate the global or other agent's state. And in a MAS, validating individual agent does not guarantee the validity of the system.

- Another runtime monitoring approach is log analysis. In this approach, all individual agents are responsible to write their current state to a log file, at prespecified events. This log file can then be used to analyze the occurrences of events, or even to visualize the complete flow of execution.

The advantage of log analysis over conditional statements is that it is compatible with distributed systems. As the log files of an individual agent can also be analyzed by comparing it to the log files of other agents. Furthermore, third party statistical tools like R can also be readily used.

5. Conclusion

This paper discussed an actively researched topic of multi-agent systems, with a focus on its performance issues. Basic notions related to the topic were also explained, like software agents, multi-agent systems, artificial intelligence and distributed artificial intelligence.
Factors and indicators which affect the performance of a MAS were also discussed, along with the techniques used to test and validate them.

6. References

[1] Nwana,H. S.,Software Agents: An Overview, Knowledge Engineering Review, vol. 11, no 3, Sept. 1996, pp. 1-40.
[2] Russell S and Norvig P: 'Artificial intelligence: a modern approach', Prentice Hall (1995)
[3] FIPA Agent Management Specification, 2004. http://www.fipa.org/specs/fipa00023/
[4] Nwana. H, "Software agents: An overview," Knowledge and Engineering Review, vol. 11, no. 3, 1996
[5] Jennings, N.R., Sycara, K. and Wooldridge, M. (1998) A Roadmap of Agent Research and Develo-pment. International Autonomous Agents and Multi-Agent Systems, 1, 7-38.
[6] Franklin, S. and Graesser, A. (1997) Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents, In: Müller, J.P., Wooldridge, M.J. and Jennings, N.R., Eds., Intelligent Agents III Agent Theories, Architectures, and Languages, Springer, Berlin Heidelberg, 21-35.

[7] Nikos Vlassis, “A Concise introduction to multiagent systems and distributed artificial intelligence,” Synthesis Lectures on Artificial Intelligence and Machine Learning, 1st edition, 2007

[8] Andreas S. Jensen, “Multi-Agent Systems: An Investigation of the Advantages of Making Organizations Explicit”, 2010

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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

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

[16] Hillol Kargupta, I1ker Harnzaoglu, Brian Stafford: Scalable, Distributed Data Mining Using an Agent Based Architecture, Proceedings of High-Performance Computing, 1997.

[17] Kasper Hallenborg: Core Design Pattern for Efficient Multi-Agent Architecture, International Book Series "Information Science and Computing", pp 29-36, 2004.

[18] Y. Cao, W. Yu, W. Ren and G. Chen, "An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination," in *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427-438, Feb. 2013. doi: 10.1109/TII.2012.2219061

[19] Hannoun M., Sichman J.S., Boissier O., Sayettat C. (1998) Dependence Relations Between Roles in a Multi-Agent System. In: Sichman J.S., Conte R., Gilbert N. (eds) Multi-Agent Systems and Agent-Based Simulation. MABS 1998. Lecture Notes in Computer Science, vol 1534. Springer, Berlin, Heidelberg

[20] Wooldridge and Michael, An introduction to multi agent systems, John Wiley & Sons Ltd, 2002

[21] Wooldridge, Michael, Michael Fischer, Marc-Philippe Huget and Simon Parsons,
Model checking multi-agent systems with Mable,
In *Proceedings of the first international joint conference on autonomous agents and multiagent systems (aamas ’02:)*

[22] Ou-Yang C., Juan YC., Li C. (2009) Applying Petri Net to Analyze a Multi-Agent System Feasibility - a Process Mining Approach. In: Chou SY., Trappey A., Pokojski J., Smith S. (eds) Global Perspective for Competitive Enterprise, Economy and Ecology. Advanced Concurrent Engineering. Springer, London

[23] Margaris, A.I. Int J Parallel Prog (2009) 37: 195. https://doi.org/10.1007/s10766-009-0093-x

[24] Parasumanna Gokulan, Balaji & Srinivasan, D. (2010). An Introduction to Multi-Agent Systems. 10.1007/978-3-642-14435-6_1.