Journal of Medical Systems (2019) 43: 274
https://doi.org/10.1007/s10916-019-1411-5

SYSTEMS-LEVEL QUALITY IMPROVEMENT

Implementation of Software Agents and Advanced AoA for Disease Data Analysis

K. Vijayakumar[1] · K. Pradeep Mohan Kumar[2] · Daniel Jesline[3]

Received: 2 April 2019 / Accepted: 26 June 2019 / Published online: 6 July 2019

Abstract

To eliminate the possibilities of getting various contradicting solutions to a single problem during diagnosis, a single regular Agent oriented Approach (AoA) is replaced by Intelligent Artificial Agents that act like human and even dynamically decide in any situations known as Intelligent Searching Approach (ISA) is proposed. These agents are used to analyse the medical forums and results or findings are derived accurately than any manual approach. Multiple Agents have been used to analyse the blogs by dividing the work areas and communicating themselves using Agent Communication Language (ACL) and FIPA. The local solutions thus formed are forwarded to a global agent. This Global Agent controls all operations and makes the decision about the best solution. As the Global Agent controls all other agents, it eradicates unwanted and ineffective communication between the various local agents and hence keeping the time taken for communication at the minimum level. Based on these solutions a prioritization matrix is formed using advanced clustering techniques to create a prioritized content of suggested best solutions. Once the decision is made, the refining process runs several times recursively checking for all possible better solutions solving the input. On completion of this process, the Global Agent returns the exact result of the discussion. This process saves time rather than researching the entire blog for result data. This advanced approach lights a different way of obtaining solution keeping the time taken for discussion and intercommunication between the agents to the minimal level but not compromising on the perfection of the solution at the same time.

Keywords Disease data analysis · Software agents · Intelligent searching approach

Introduction

An agent based model or ABM serves the purpose of conducting several simulations to find out the reactions of both separate parties as well as grouped networks under various conditions, so that a thorough understanding of the system under study can be obtained. This model functions well due to the combined strategies of several sub-fields such as socio-economic computation, parallel processing and human developmental studies. The approach of using trial-agents has always been disussed in several branches of programming science, under various contexts. In the field of Artificial Intelligence (AI) or Distributed AI, these trial methods are the typically exploited techniques to handle complex problems and to develop intelligent software systems [1–3]. Aarchetypes of Agent-Oriented Programming dates back to the first paper written in 1993 [4], and since then many Agent Programming Languages (APL) and languages for Multi-Agent Programming have been proposed [5–7]. The objective of Agent Oriented Programming, as introduced in 1993 was considered to be the meaning of a post Object-Oriented programming paradigm for the development of high-performance applications, provision of complex level features compared to the existing prototypes.

Without considering real-time machine workloads process plans may become suboptimal or even invalid at the time of

This article is part of the Topical Collection on *Systems-Level Quality Improvement*

✉ K. Vijayakumar
mkvijay@msn.com

K. Pradeep Mohan Kumar
kpratheep15@gmail.com

Daniel Jesline
jezline@gmail.com

[1] Department of Computer Science &Engineering, St. Joseph's Institute of Technology, Chennai, India

[2] Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India

[3] Department of Computer Science &Engineering, St. Joseph's of College of Engineering, Chennai, India

execution. Therefore, there is a need for the integration of process-planning and scheduling systems for generating more realistic and effective plans [8].

Even though there seems to be no clear definition of the word "agent", one thing most definitions tend to have in common, is the view that any individual element (software, model, individual, etc.) can be referred to as an agent (Bonabeau 2001).

Establishing loosely coupled collaborations between services provided by autonomous enterprises poses several requirements and challenges for the surrounding service ecosystem. In such context emphasis must be laid especially on the correctness of available metainformation and its usage. Towards this purpose, we characterize service ecosystems by a set of metamodels [9].

Feature selection is the approach of choosing subset of given dataset based on some feature. It can be used to minimize dimensions of the huge data set. So that it removes unnecessary data in the data source and produces prediction or output accurately in big data analytics. In the proposed work, feature selection algorithm process is implemented for text categorization using the algorithms ant colony optimization (ACO) and artificial neural network (ANN). This hybrid approach simulated using Reuter's data set and proved its efficiency [10].

The proposed paradigm permits the client to authenticate the validity of the encrypted secret key produced by the TPA for uploading data to the cloud [11].

The main aim of this system is to constantly scan the applications that are deployed in the cloud and check for vulnerabilities as part of the continuous integration and continuous deployment process. This process of checking for vulnerabilities after every update in the application should be included in the software development lifecycle [12].

Security aspects of the software solution depends on the code which needs to be consistently tracked on a continuous basis to monitor changes and its related risks, without which vulnerabilities and weakness identification will be reactive. It is always essential to identify risks based on the experience from others that is where the use of risk assessment frameworks would be handy along with vulnerability and weakness database such as common weakness enumeration, common vulnerability enumeration and Exploit DB [13].

Advanced agent oriented approach

The whole process in this paper is controlled and implemented using AoA. The Main agent controls all other agents and all the activities associated with it. The Fig. 1 explains the overall process in a nutshell.

There are three algorithms in the proposed work and these algorithms are controlled by the main algorithm which acts as the brain of the main agent which controls the overall process. The detailed description of the algorithms is as follows.

The Main Agent algorithm focuses basically on invoking these three algorithms at the necessary time and also handling the recursive calls. It also involves in steps that ensure its own security like synchronization ensuring process non-kill scenarios and so on. The definitions of all the four algorithms are given below.

Preprocessing/Data Retrieval Layer

I/p: Unprocessed Data
Process: Arrange Data in order/format check the states of all the agents and check memory buffer
O/p: Pre-processed Data, All agents OK status, memory – GO status

1. Initialize Prioritization Matrix to 0,0 for all solution sets.
2. Initialize Possible_Solutions_Suggested Matrix to 0,0 for all solution sets.
3. 3.InititateBest_SolutionsMatrix to 0,0 for all solution sets.
4. Initiate Check_State Agent.
5. Check the state of all agents by sending a test message.
6. Expect a reply from all the agents with the status report as Sleep/Awake.
7. If (No Reply from any agent).
8. Create an error report and alert Global Agent.
9. Return Error_Status code to Global agent terminate Algorithm 1.
10. else Check the status and memory buffer of the Global Agent and Local Agents.
11. If (OK) continue else, Free the memory of Global Agent and Local Agents.
12. Check the format of the input files.
13. If (The Inputs are in proper Tree Format) continue the process.
14. Else Initiate Topic_Decider Agent.
15. Search for the most appearing and most important terms.
16. Create default topics using these terms and add any number of discussions under the new topics.
17. If (Any left out discussion thread).
18. Form a topic named Anonymous and add these discussions in this topic.
19. Return Status_Code as OK to Main Agent.
20. End.

This Algorithm is completely dedicated for pre-processing and retrieving of data. It also focuses on the initial conditions set for the smooth operation of the different agents and keeps a note on the memory to be used by the agents. Create a few initial matrices for the processing of the Algorithm group like

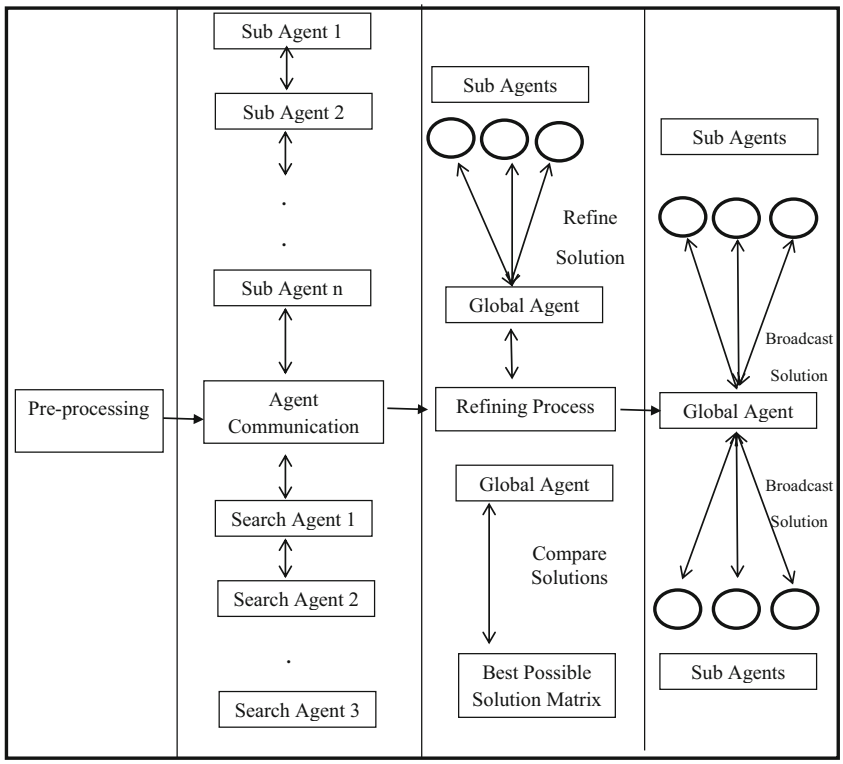

Fig. 1 Overall System Architecture of AoA

Prioritization Matrix, Possible_Solutions_Suggested Matrix, Best_SolutionsMatrix. A dedicated agent called check agent is used to check the whole status of the process is a go or not. It checks the agents by broadcasting a ping message and expects a reply from all the agents. With this, it can form a list of agents that are sleep and awake. If there is any error, it notifies the Global Agent/Main Agent and terminates. This algorithm also checks the Data Warehouse/Live Data and does a few initial pre-processing steps like diving the areas of search, giving some topics to them, allocating few local agents for individual search areas. It returns the status to the Global Agent when called.

Agent communication for prioritisation / data processing layer

I/p: Search Term
Process: Query Management
O/p: Local Solutions

1. Divide the Data Sets/Live Data into smaller chunks
2. If (Complex Topics with more than 500 Discussion Threads)
3. Divide the Complex Discussion Thread as sub Agent Search areas
4. else Allocate one Local Agent for every discussion topic
5. Initiate the Search_Topic Procedure
6. Compare the search term with the discussions and come up with Local Solutions
7. Use ACL Message Format to intercommunicate between the Local Agents
8. Check the message content code
9. If(1) The Target Local Agent wants to ask a Question to the current Local Agent
10. If(2) The Target Local Agent wants to propose a better solution to the current Local Agent
11. If(3) The Target Local Agent wants to propose a better solution to the current Local Agent's proposal earlier and object its solution
12. If(4) The Target Local Agent wants to object the solution of the current Local Agent
13. If(5) The Target Local Agent agrees the solution of the current Local Agent
14. If(6) The Target Local Agent wants to communicate it has no solution to the current Local Agent
15. If any confusion, report to the Global Agent
16. Start the Finalize_Communication procedure
17. Use KQML to finalize communications and return the Local Solutions in a prioritization matrix to the Global Agent
18. End

This Algorithm concentrates on the Agent communication and the underlying data processing. Divide and subdivide the Data set/Live Data and allocate a few agents for the subtopics.

It also discusses a few codes on what an agent wants to communicate to the other agent. This eliminates the confusion between agents and reduces the overload of any text data for communication. If there is any discrepancy, the Global agent takes care of the problem. It returns the Local Solutions that are formed by the local Agents in a prioritized matrix format to the Global Agent.

Refining Process/ Quality Enhancement Layer

I/p: Input Search term, Local solutions formed by the Global Agent.
O/p: Possible_Solutions_Suggested matrix.
Process: Correctness of Solution – Query Re evaluation.

1. Collect the Local Solutions of the sub- agents and Local Agents.
2. Form a Global Solution Set.
3. Take the maximum repeated solutions and refine the Global Solution set.
4. Broadcast the probable best Global Solution.
5. If (Any Local Agent/Sub-Agent has a better solution than this proposed Global Solution).
6. Reply with the proposed solution.
7. Replace the value of the possible Global Solution with the new value proposed.
8. Repeat step 4.
9. Else.
10. No Agents must reply (ensures minimum communication).
11. Return the possible Global Solution as Possible_ Solutions_Suggested matrix.

This Algorithm concentrates on the refining process and selecting the possible solutions. It collects all the local solution sets from the agents and sub agents. With the help of these, it forms a Global Solution set. Refine the process by prioritizing the solutions that are repeated more times. Broadcast the possible solution set to all the Agents and subagents and then it waits for the reply of the local agents. If any local agent has a better solution, it may reply. To minimize the communication levels, the local agent is restricted from replying or suggesting if it does not have any solution better than the broadcasted one. This algorithm returns back a matrix with the most possible solutions.

Main Agent: Base Processing Layer

I/p: Search term
O/p: Global Solution
Process: Controlling all Agents and communication between Agents, Compute Global Solution

1. Main Agent starts the process of Algorithm 1(Pre-processing on the Data Sets/Live Data)
2. Check the Status_Code, If (OK) continue else terminate the algorithm with an error message
3. Initiate I/p Thread
4. I/p Thread gets the forum Question
5. Categorize two possibilities – May and May not based on the topics of the Data Sets/Live Data
6. If (Case: May) //Question Present in the Forum
7. May Thread initiated which in turn initiates subagents to start the process
8. Start the Search Answer procedure
9. Initiate multiple sub agents to search in their allotted area
10. Form Local Solutions
11. Initiate prioritize procedure (Algorithm 2) and get back the value of returned prioritization matrix
12. Call the refining procedure (Algorithm 3) get back the value of returned Possible_Solutions_Suggested matrix
13. Initiate the Global Decision maker Agent
14. Compare the Local Solutions obtained by the main agent with the prioritization matrix and store the output as Best_Solutions
15. Compare the Best_Solutionswith the Possible_ Solutions_Suggested matrix
16. Decide the best possible Global Solution
17. Else (Case: May not)// Question not present in the target Data Set/Live Data
18. Import all the available Data sets/Live Data and include all the Topics
19. Start the find Answer procedure
20. Start the Digging Agent
21. Dig all possible solution using the digging agent across various topics of the Data Set/Live Data
22. Initiate correction procedure
23. Predict/ Find the area the User is trying to dig and store the area as Target Area
24. Remove the results of the Digging Agent from all other areas
25. Use the data of recently viewed, recently updated, most viewed, commented, most liked to improve the search results and form the Local Solutions
26. Initiate prioritize procedure (Algorithm 2) and get back the value of returned Prioritization matrix
27. Call the refining procedure (Algorithm 3) get back the value of returned Possible_Solutions_Suggested matrix
28. Initiate the Global Decision maker Agent
29. Compare the Local Solutions obtained by the main agent with the prioritization matrix and store the output as Best_Solutions
30. Compare the Best _Solutions with the Possible_ Solutions_Suggested matrix and decide the best possible global solution

The overall process associated by combining all these algorithms is explained in the process flow diagram in Fig. 2.

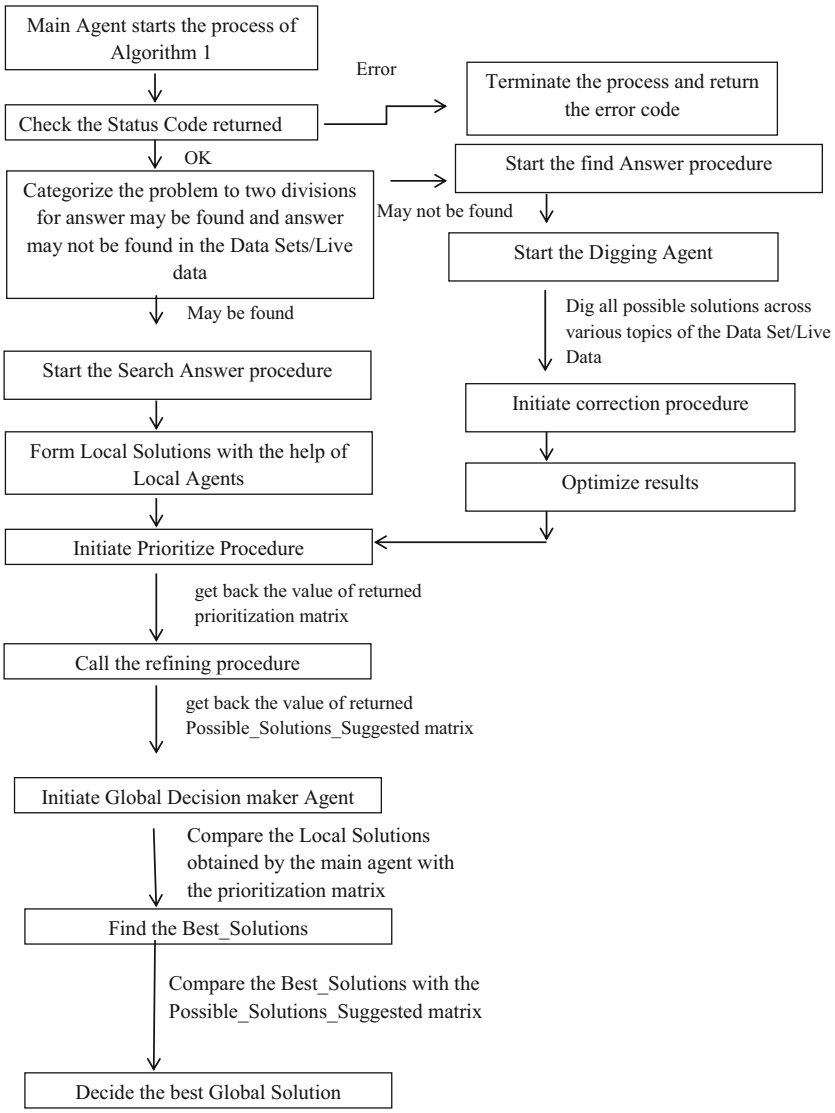

Fig. 2 Flowchart of AoA approach

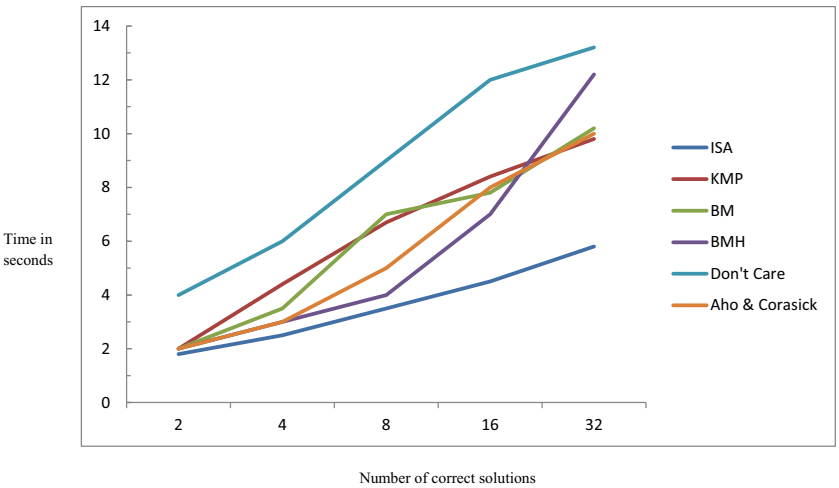

Fig. 3 Algorithm Stability

Table 1 Algorithm Stability

Time (in seconds)	ISA	KMP	BM	BMH	Don't Care	Aho&Corasick
2	1.8	2	2	2	4	2
4	2.5	4.4	3.5	3	6	3
8	3.5	6.7	7	4	9	5
16	4.5	8.4	7.8	7	12	8
32	5.8	9.8	10.2	12.2	13.2	10

Results & Discussion

Algorithm Stability

Ability to get the perfect solution in minimum time is called as Algorithm Stability. The following graph shows the stability of different algorithms compared with the proposed algorithm. Figure 3 and Table 1.

This table demonstrates the power of the Advanced AoA over other famous and competitive algorithms implemented in the same filed. The time taken to achieve correct solutions over numerous data sets is efficient and effective using Advanced AoA. The detailed explanation of the comparative study is given below.

Implementation Tools used:

The implementation tools that are used to apply the algorithms on the given data set/Live Data are as follows

- JADE version 4.3.3 from http://jade.tilab.com/
- Netbeans 8.0.1 embedded with JADE GUI,JUnit and JTest

Conclusion

To give solutions to a variety of business and technology related problems, agent based models have been in use since the mid-1900s. Supply chain optimization and logistics, social network effects, workforce management, distributed computing, modelling of consumer behaviour, including word of mouth, and portfolio management are all examples of its application. Analysis of traffic congestion is yet another application. The system of interest is simulated in these applications by capturing the behaviour of the individual agents as well as the interconnections among them. Agent-based modelling tools can be put to use to help test how the changes in individual behaviours will affect the system's emerging overall response. The agent based evolutionary search is now a new research topic for solving complex problems that deal with optimization.

Compliance with Ethical Standards

Conflict of Interest We declare that there is no conflict of interest in this research article.

Research Involving Human Participants and/or Animals This research article does not contain any human participants or animals performed by the authors.

Informed Consent No humans are involved in this research article.

References

1. Jennings, N. R., An agent-based approach for building complex software systems. Commun. ACM 44(4):35–41, 2001.
2. S. Russell and P. Norvig. Artificial Intelligence, A Modern Approach (second edition). Prentice Hall, 2010.
3. M. Wooldridge. An Introduction to Multi-Agent Systems. John Wiley & Sons, Ltd, 2002.
4. Shoham, Y., Agent-oriented programming. Artificial Intelligence 60(1):51–92, 1993.
5. R. H. Bordini, M. Dastani, J. Dix, and A. El FallahSeghrouchni. Special Issue: Multi-Agent Programming, volume 23 (2). Springer Verlag, 2011.
6. R. H. Bordini, M. Dastani, and A. El FallahSeghrouchni, editors. Multi-Agent Programming Languages, Platforms and Applications - Volume 1, volume 15.Springer, 2005.
7. R. H. Bordini, M. Dastani, A. El FallahSeghrouchni, and J. Dix, editors. Multi-Agent Programming Languages, Platforms and Applications - Volume 2.Springer, 2009.
8. Shen W, Wang L, Hao Q. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. IEEE Transactions on Systems, Man, and Cybernetics, PartC2006; 36(4):563–77.
9. Ruokolainen, T., Kutvonen, L.' Managing interoperability knowledge in open service ecosystems' In: Proceedings of the 13th Enterprise Distributed Object Computing Conference
10. R. Joseph Manoj, M.D.AntoPraveena, K. Vijayakumar, "An ACO–ANN based feature selection algorithm for big data", Cluster Computing The Journal of Networks, Software Tools and Applications.
11. K. Vijayakumar, S. Suchitra and P. SwathiShri, "A secured cloud storage auditing with empirical outsourcing of key updates", Int. J. Reasoning-based Intelligent Systems, Vol. 11, No. 2, 2019.
12. K. Vijayakumar C. Arun, Continuous security assessment of cloud based applications using distributed hashing algorithm in SDLC, Cluster Computing Springer 2017.
13. Vijayakumar, K., Arun, C., Automated risk identification using NLP in cloud based development environments, J Ambient Intell Human Computing, Springer 2017.