Combining Multi-Agent Systems and Subjective Logic to Develop Decision Support Systems

César González-Fernández(✉), Javier Cabezas, Alberto Fernández-Isabel, and Isaac Martín de Diego

Data Science Laboratory, Rey Juan Carlos University, c/ Tulipán, s/n, 28933 Móstoles, Spain
{cesar.gonzalezf,javier.cabezas,alberto.fernandez.isabel, isaac.martin}@urjc.es
http://www.datasciencelab.es

Abstract. Nowadays, the rise of the interconnected computer networks and the increase of processed data have led to producing distributed systems. These systems usually separate multiple tasks into other simpler with the goal of maintaining efficiency. This paradigm has been observed for a long time in different animal organisations as insect colonies and fish shoals. For this reason, distributed systems that emulate the biological rules that govern their collective behaviour have been developed. *Multi-Agent Systems* (*MAS*) have shown their ability to address this issue. This paper proposes *Ant Colony based Architecture with Subjective Logic* (*ACA-SL*). It is a bio-inspired model based on ant colony structures. It makes use of *MAS* to distribute tasks and *Subjective Logic* (*SL*) to produce *Decision Support Systems* (*DSS*) according to the combination of individual opinions. A system implementation based on the proposed architecture has been generated to illustrate the viability of the proposal. The proposed architecture is intended to be the starting point for developing systems that solve a variety of problems.

Keywords: Multi-Agent system · Subjective Logic · Bio-inspired system · Distributed organisation · Decision Support System

1 Introduction

In recent times, the heyday of the Internet and the advance of technology have produced tons of data which are processed by several systems [1]. These systems apply the strategy of separating data into simpler and smaller pieces in order to process them efficiently. Thus, the information extraction task and the generation of knowledge are simplified. This issue has led to the resurface of distributed systems.

Multi-Agent Systems (*MAS*) [2] are a specific case of distributed systems. They use agents that are software abstractions able to perform tasks and to satisfy the associated goals interacting with the environment around. These agents

M.-J. Lesot et al. (Eds.): IPMU 2020, CCIS 1237, pp. 143–157, 2020.
https://doi.org/10.1007/978-3-030-50146-4_12

present desirable features as: autonomy [3], flexible behaviour to react to changes in the environment in a timely fashion [4], and dynamic interaction between them [5].

Nevertheless, these systems have some shortcomings. The organisation to solve specific situations is one of the most typical challenges [6]. In this regard, bio-inspired mechanisms are one of the most used self-organisation solutions [7]. Thus, they can adapt some social animal behaviour to solve specific situations. Typical instances of these mechanisms are insect colonies [8], fish shoals [9] and mammals packs [10], where the solution of a complex problem is achieved by the individuals solving simpler issues.

This paper proposes *Ant Colony based Architecture with Subjective Logic* (*ACA-SL*), a novel architecture based on bio-inspired *MAS* and *Subjective Logic* (*SL*) [11] to develop distributed *Decision Support Systems*(*DSS*) [12]. These latter are able to produce evaluation according to a specific topic or domain according to a previously obtained knowledge. *ACA-SL* has been developed as part of the SABERMED project, which is funded by the Spanish Ministry of Economy and Competitiveness. *ACA-SL* emulates the behaviour of ant colonies to execute distributed jobs. The way in which jobs are defined exhibits a high degree of flexibility for multiple application scenarios. For this purpose, there will be several agents assuming the same role as workers in ant colonies. Jobs are defined as the combination of very diverse tasks which may hold some dependencies among them. This fact provides agents with the capability to work on the same job at the same time. The architecture combines the opinions generated by following the rules, and methodologies defined in the *SL*.

A system based on *ACA-SL* has been implemented to show the viability of the proposal. It has been used to analyse websites and generate an opinion concerning the degree of trust applicable to them. The created opinions are the result of processing related information extracted from the websites under analysis (e.g. domain and Whois).

The remainder of the paper is structured as follows. Section 2 situates the proposal in the domain and make comparisons with previous approaches. Section 3 details *ACA-SL* and its components. Section 4 presents the experiments. Finally, Sect. 5 concludes and proposes future guidelines.

2 Background

This section introduces the foundations of *ACA-SL*. It overviews the concept of *MAS* (see Sect. 2.1), both by defining it and also by providing some details on the current state of art. Secondly, insect colonies and their internal organisational procedures are introduced (see Sect. 2.2). Finally, *SL* foundations and most common applications are presented (see Sect. 2.3).

2.1 Multi-Agent Systems

Agents can be defined as intelligent autonomous entities able to act, partially perceive the environment they live in, interact with it and communicate with

other agents [13]. They take part in an organised activity in order to satisfy the particular goals they were designed to, both by executing a set of skills and by interacting with other agents. These goals are evaluated by a mental state. The mental state acts as the brain of the agents, containing steps and rules. Therefore, agents show pro-activeness (they exhibit goal-directed behaviour by taking initiative), reactivity (they perceive their environment and respond in a timely way to changes that may occur in the environment) and social awareness (they cooperate with other agents in order to accomplish their tasks) [14].

MAS [15] are a loosely coupled set of agents situated in a common environment that interact with each other to solve complex problems that are beyond the individual capacities or knowledge of each agent [16]. These systems are found in a wide spectrum of heterogeneous applications such as simulations [17], optimisation problems [18] and computers games [19]. *MAS* have been also used in the literature with the purpose of distributing very demanding tasks [20]. They are able to use agents that perform simple tasks in order to generate a more complex and demanding one. Fields of application where these kind of approaches are considered are road traffic [21] and communication networks [22].

There are multiple frameworks available to implement software based on *MAS*. Many of them respond to the restless evolution and unremitting development occurring both in the industry and in the scientific community. JADE (Java Agent Development framework) [23], FIPA-OS (Foundation for Intelligent Physical Agents Operating System) [24] and SPADE (Smart Python Agent Development Environment) [25] exemplify some of the existing options at disposal of the user.

MAS can be designed by using Agent-Based Modelling (ABM) [26] and Agent-Oriented Software Engineering (AOSE) techniques [27]. These ones are considered by solid methodologies to simulate relationships and communication channels between agents. Instances of well-known agent methodologies are INGENIAS [28] and Tropos [29].

ACA-SL models a *MAS* that identifies agents as workers belonging to an ant colony. Notice that at this point, these workers are only modelled through a finite state machine, instead of defining explicitly goals and mental states. The implementation achieved to validate the proposal has been developed using the SPADE framework.

2.2 Insect Colonies

Many species of social insects exhibit the division of labour among their members. This behaviour can be observed in bumblebee colonies [30], termites colonies [31] and wasp colonies [32]. The specific task allocation can be determined by multiple features. The age of the individual [33], the body size [34] or the position held in the nest [35] are some instances of these features. Several works concentrate on these behaviours in order to propose new task allocation strategies in artificial systems [36].

Regarding the task allocation method used by individuals, it can be modelled by using response thresholds [37]. These thresholds refer to individual tendency

to react to task-associated stimuli. For the specific case of ants, it is considered that every task can exert certain level of influence over them. Thus, if the stimulus that a task exerts on an ant is higher than its response threshold, the ant engages to this task. This leads to considering the existence of castes in which individuals may have different response thresholds. In the case of artificial systems, the use of these response thresholds to solve labour division have been used to enhance response times and load balancing issues [38].

ACA-SL uses a model based on two different types of ants according to a specific response threshold. The architecture provides a specific definition to the measure of the stimuli and the response threshold level.

2.3 Subjective Logic

SL [11] is a type of logic that allows playing with subjective beliefs. These ones are modelled as *opinions*. The *opinions* represent the probabilities of a proposition or an event with a certain degree of uncertainty. *SL* defines a set of operations that can be applied to the *opinions*. Typical instances of these operations are: *addition*, *subtraction*, *cumulative fusion* and *transitivity*.

SL extends the traditional belief function model [39]. This logic is also different from Fuzzy logic [40]. While Fuzzy logic uses vague propositions but provides accurate measurements, *SL* works with clear propositions and uncertain measures.

Given a binomial variable ($true$ or $false$) representing a proposition x, and a source of *opinions* A, an *opinion* provided by A about x, w_x^A, is represented by a quadruple of values as follows:

$$w_x^A = \{b_x, d_x, u_x, a_x\}, \tag{1}$$

where b_x is the mass belief (belief supporting that x is $true$), d_x is the disbelief mass (belief supporting that x is $false$), u_x is the uncertainty mass and a_x is the atomicity rate.

Regarding the features of *SL*, they have made this logic suitable for applying it to multiple projects covering different knowledge areas. Thus, in general, it can be used to build frameworks for *DSS* [41]. More specifically, *SL* can be used in Trust Network Analysis to calculate the trust between different parts of the network where trust measures can be expressed as beliefs [42]. Analogously, in mobile networks, *SL* can be used to calculate the reputation of the communication nodes [43]. *SL* can be also used in applications independent from the technology (for instance, legal reasoning [44]).

The proposed architecture makes use of *SL* to handle the beliefs that can be generated as a result of the different tasks processed. These beliefs are modelled as *opinions* using only a specific subset of operations.

3 Proposed Architecture

This section details *ACA-SL*. The aim of this architecture is to produce a design to develop *DSS* able to make evaluations. It combines bio-inspired *MAS*

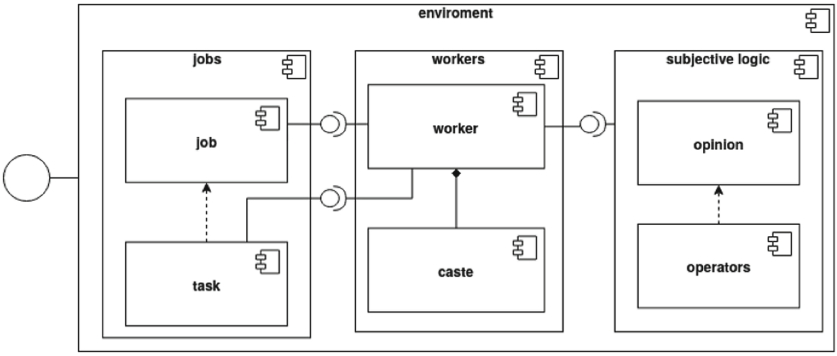


Fig. 1. Components defined by *ACA-SL*.

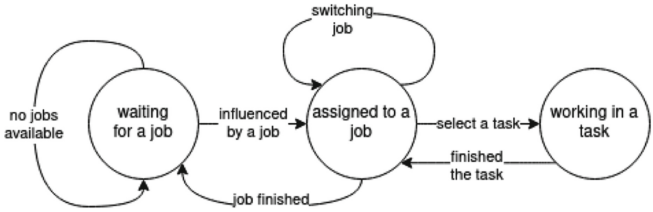


Fig. 2. Life cycle of a worker as a finite state machine.

approaches with *SL* to achieve this issue. Overall jobs are decomposed into multiple tasks which are assigned to the different agents. Agents take full responsibility on a successful accomplishment of the assigned tasks. Notice that *ACA-SL* defines the baselines on how the jobs should be divided into atomic tasks. Individual results arising from their internal processes are then combined to obtain a solution for the global problems.

Figure 1 shows an overview of the proposed architecture. A system based on this architecture takes responsibility on executing the jobs. These ones correspond to needs that the external systems may require to satisfy.

Next sections address the internal procedures followed by agents, detailing jobs and their inner structure. They also describe how *SL* is implemented in the proposal.

3.1 Multi-Agent System Based on Ant Colonies

Analogous to nature, the proposed architecture presents an environment where workers live in. The behaviour of workers is represented by a finite state machine with three states (see Fig. 2).

Delving into the behaviour of workers, the registering of a new job represents a change happening in the environment. These changes (i.e. new jobs) exert stimuli that are perceived by workers, which can be influenced by it. To prevent

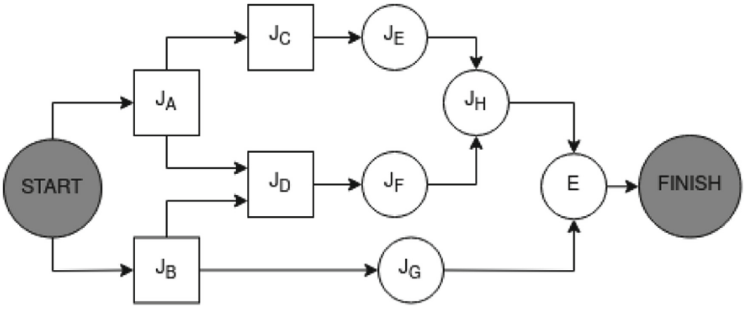


Fig. 3. Instance of a job graph and its tasks.

that some task remains in inconsistent states, only idle workers (those not running any task) are influenced by jobs. Workers make a decision on whether to take responsibility on the new jobs created based on this job influence.

Every worker presents an internal threshold which is compared to the job influence value itself to determine whether the latter presents a higher value and consequently, a switch to the new job is required. Based on this threshold, the bio-inspired approach defines two castes of workers [37]: *major* and *minor*. Those workers simulating *major* ants will show a higher threshold than the one assigned to the workers representing *minor* ants. This feature allows reserving workers to carry out specific jobs. For example, if the influence of a job is calculated based on its priority, the major workers only perform high-priority jobs. This feature plays a crucial role in systems where resources availability, response times and load balancing are critical and very demanding [45].

3.2 The Job Workflow

A job gets represented by directed graphs (see, for instance, Fig. 3). Its component tasks can be interpreted as the multiple possible road-maps linking the *start* node (i.e. starting point) with the *finish* (i.e. finishing point). The workers assigned to a job that are not running any task are placed at the *start* node. These workers are continuously checking the status of all tasks connected to the *start* node via directed edges. If all connected tasks present are being run by other worker, then workers wait at the *start* node. Those tasks connected to the *start* node that are not being executed, present themselves as potential candidates to be selected by workers.

Workers are oriented towards the task selection issue. Thus, there are measures which provide cost values to the different edges between nodes.

A job is considered to be successfully completed when the worker handling the last task represented by the *finish* node completes its duties. Notice that jobs are independent from each other.

Regarding the intermediate nodes of this graph, they represent the different tasks in which the job is divided, giving shape to multiple possible paths linking

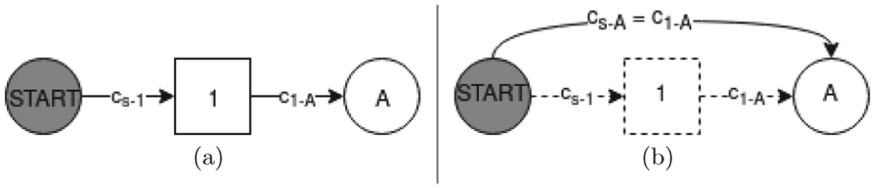


Fig. 4. a.- State of the path from (start) to (A) before task (1) is completed. b.- State of the path from (start) to (A) after task (1) is completed.

the *start* node with the *finish* node. It may be the case when there is not any path from the *start* node to the *finish* node (e.g. the intermediate nodes have raised errors). This situation translates into a job finishing with failures during execution.

Notice that the next tasks available in the path are the next related to the last completed one (i.e. a completed task is no longer visible as available tasks for workers). Figure 4 illustrates this point with an example. Let c_{s-1} be the cost associated to the edge connecting nodes *start* and (1). Let c_{1-A} be the cost between (1) and (A). Figure 4(a) shows one path and one task connected to the *start* node along with the cost involved in the different edges. When the task (1) is completed, it is hidden and the *start* node gets virtually connected to the node (A) by establishing a new edge with cost value c_{s-A} equal to the c_{1-A}.

Regarding the tasks, they are considered as atomic. Every worker assigned to a job is responsible for carrying out just one of the component tasks at a time. Hence, a one-to-one relationship between workers and tasks is established. Tasks assigned to workers contain specific prerequisites to be fulfilled. These preconditions are addressed to ensure correct alignment of workers.

According to these preconditions, tasks are organised into two main groups. The first group considers the tasks that require the fulfillment of all the prerequisites to be executed (labelled as *strict*), while the second group includes tasks executed every time a requirement is satisfied (labelled as *soft*).

In Fig. 3, the *strict* tasks are represented by squares, while the *soft* tasks are pictured by circles. In this example, task represented by node (J_D) cannot be performed until task (J_A) and (J_B) are completed. On the other hand, task (J_H) is executed when (J_E) or (J_F) are completed. On this way, the requirements can be only satisfied with the result of an individual previous task.

Tasks follow a specific workflow to manage their own internal state. Five states are defined in this regard: *waiting*, *running*, *completed*, *error* and *blocked*. Figure 5 shows the dynamic flow and possible relationships between states.

When a new job is created, all component tasks are in *waiting* state. Once a worker is in a position to start with a task (i.e. fulfillment of its particular requirements), the task changes its internal *waiting* state to *running*. A successful completion of the task results in a *completed* state for it. However, if errors were found during the process, the task changes to *error* state. Notice that any other worker can take responsibility for a task in the *error* state to seek its completion

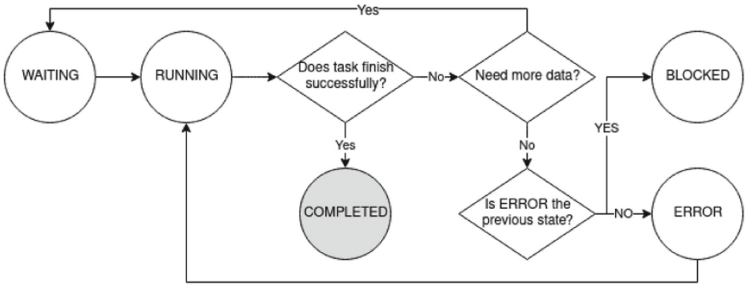
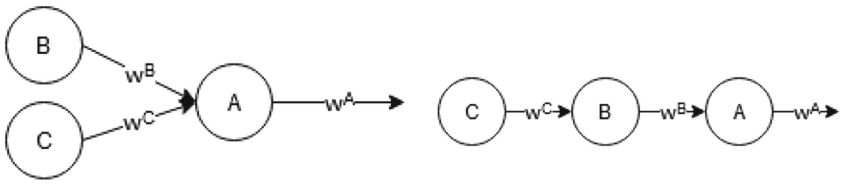


Fig. 5. Flow diagram of the states of a job.



(a) Cumulative fusion (b) Transitivity

Fig. 6. Graphical representation of the *SL* operator.

(even the same original worker). However, tasks can also enter into a *blocked* state when, after being in *error* and proceed with retrial, *completed* state is not reached. All tasks in *blocked* state are removed from the pool of available tasks for workers, which results in not considering neither their nodes nor the edges connected to them in the graph.

3.3 Combining Opinions with Subjective Logic

The proposed architecture lies in its ability to deal with beliefs. In pursue of that feature, *SL* is considered as a methodology to manage these beliefs. The belief can be the results of the execution of a task.

Considering the fact that *ACA-SL* currently finds itself at a very early stage, just binomial opinions are taken into account in the remaining of this section. Likewise, a reduced subset formed by two operators is contemplated in the proposed architecture (see Fig. 6) : *cumulative fusion operator* ($w_x^{A \diamond B} = w_x^A \oplus w_x^B$) and *transitivity operator* ($w_x^{A;B} = w_x^A \otimes w_x^B$).

The use of *SL* enables to manage opinions given by multiples sources. These opinions can be combined. Furthermore, the sources can have different robustness levels based on the confidence in each of them. The confidence in a source can be assigned by manual setup, using rules defined by human experts, or can be dynamically defined by training the system (e.g. using a previously evaluated dataset).

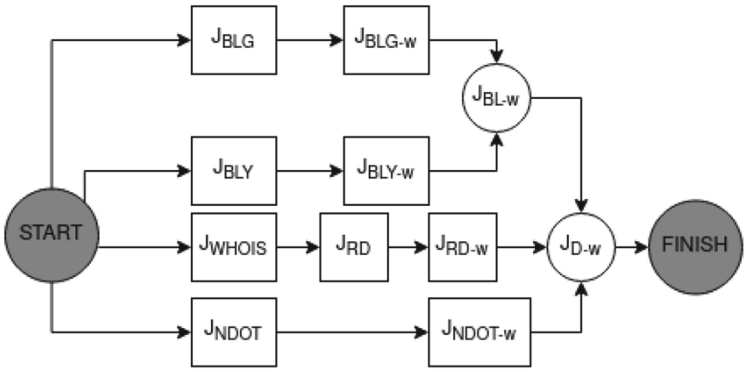


Fig. 7. Job graph produced for the experiment.

Table 1. System configurations for the proposed experiments.

	#major	#minor	major th.	minor th.
Configuration 1	0	1	–	0
Configuration 2	5	20	5	0
Configuration 3	5	20	10	0
Configuration 4	20	50	5	0

4 Experiments

A *DSS* based on *ACA-SL* has been implemented to evaluate the validity of the proposed architecture. The system purpose is to identify malicious web domains. Thus, given a domain, the system is capable of generating an opinion about it. This opinion is based on specific methods gathered from the literature of the domain. These methods are to query well-known blacklists [46], to check both the number of dots in the domain [47] and the registration date of the domain [48].

A job that includes the specific methods has been created to evaluate a domain. Figure 7 shows the graph of the implemented job. This job is divided into multiple tasks. (J_{BLG}) and (J_{BLY}) tasks query the blacklists of *Google* and *Yandex* respectively. (J_{WHOIS}) obtains the Whois, while (J_{RD}) extracts the registration date from the Whois data, and (J_{NDOT}) obtains the number of dots in domain. The objective of these tasks is to retrieve information about the domain. This information is then used by the following tasks to generate opinions. (J_{BLG-w}) and (J_{BLY-w}) give an opinion based on blacklists responses, (J_{BL-w}) combines preceding opinions, (J_{RD-w}) generates an opinion about the registration date, (J_{NDOT-w}) use the count of dots in the domain to give the opinion and, finally, (J_{D-w}) combine all of these opinions to provide a final resulting opinion about the domain.

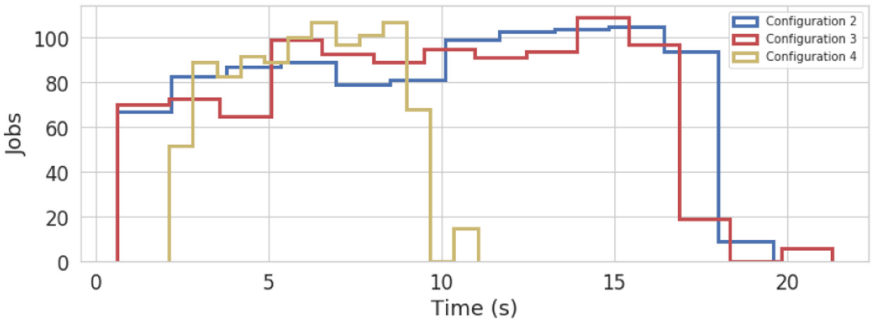


Fig. 8. Time consumed by the jobs.

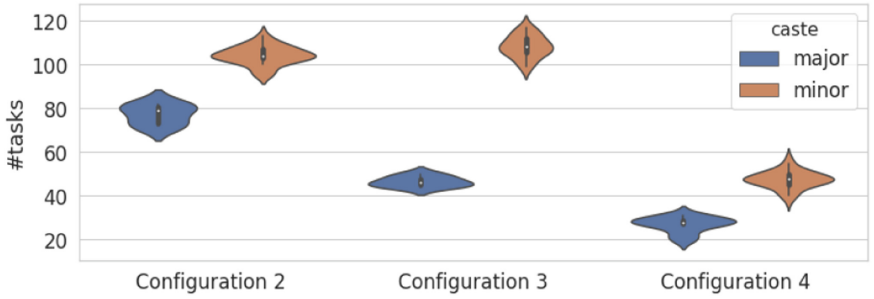


Fig. 9. Number of tasks carried out by each worker.

Four configurations of workers have been tested. Table 1 shows the parameters for each experiment. *#major* and *#minor* indicate the number of ants belonging to each category, while *major th.* and *minor th.* reflect their respective thresholds.

The system has also been customised according to specific settings. First, when the influence of multiple jobs exceeds the threshold of a worker, the worker selects the job with the greatest influence. Secondly, Eq. 2 is used to calculate the influence exerted by a job (I_j). This influence is proportional to the age of the job ($T_j(s)$) (i.e. the current time subtracting the initial time the job is registered in the system) and inversely proportional to the square of the number of workers (W_j^2) assigned to it. To avoid a potential division by zero, one is added to the denominator:

$$I_j = \frac{T_j(s)}{W_j^2 + 1} \tag{2}$$

In this configuration, the two blacklist methods are preferred over the rest ones. To indicate this preference, the cost of the edges used to form the paths passing through these tasks is set up with lower values. Finally, given a domain, not appearing in a blacklist is not sufficient to consider it as trustworthy.

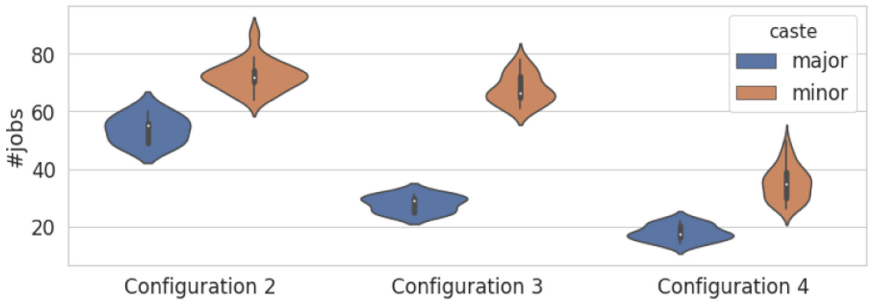


Fig. 10. Number of times each worker change its assigned job.

To test the application, 1000 domains have been processed. For each domain, a job is registered in the system. Figure 8 shows the number of jobs and their time consumed until completion according to the selected configurations. Configuration 1 is not depicted because the time consumed is one order of magnitude higher. Times spent in Configurations 2 and 3 are similar (their jobs take between 1 and 22 seconds to finish). Configuration 4 obtains shorter times. This result illustrates that the use of several agents had a positive effect in the performance of the system.

Figures 9 and 10 depict the number of tasks and the number of job changes each worker performs in both castes respectively. In all configurations, the major workers perform less tasks than minor workers. Also, the Configuration 3 shows a larger gap between castes than the rest of the configurations. This fact is a consequence of the configured thresholds.

This experiment shows how a good selection of the configuration parameters improves the performance. The time consumed by the jobs varies drastically depending on the number of workers available. This fact indicates that the job division and the use of multi-workers are suitable. Finally, the use of different thresholds has provoked that the number of tasks performed by a major worker decreased. If needed, the resources associated to these workers can be reserved by adjusting these thresholds.

5 Conclusions

This paper introduces *ACA-SL*, a bio-inspired architecture based on ant colony structures. It combines *MAS* and *SL* to correctly manage high distributed *DSS*.

The architecture makes use of agents taking the role of ant workers. They deal with registered jobs which are external requests placed to the colony. To facilitate the parallel processing, these jobs are defined as a set of tasks which are individually assigned to the different workers. These tasks make up a graph defining the job. Finally, *SL* is used to handle the opinions generated during the process.

A basic application has been implemented to validate the proposal. Experiments have been carried out to illustrate that the proposed architecture is truly feasible. They have shown the importance of defining appropriate settings (i.e. the edge costs, the job influence equation or the job graph shape).

ACA-SL is in an early stage of development. However, foundations followed during its design are addressed to establish a good basis for future implementations. Some instances exemplifying these aspects can be found in the capability to setup internal parameters to improve offered performance and the division of jobs in tasks to guarantee correct parallel processing and resources management. Some future works will arise from this contribution. In order to facilitate future implementations based on the proposed architecture, a complete framework development is being considered. This framework will follow the ABM methodology and the Model Driven Architecture (MDA) guidelines. There is a plan to extend some of the features already defined, and to increase the number of castes with the purpose of improving the flexibility of the system. New *SL* operators will be also considered in future projects.

Acknowledgments. Research supported by grant from the Spanish Ministry of Economy and Competitiveness, under the Retos-Colaboración program: SABERMED (Ref: RTC-2017-6253-1) and the support of NVIDIA Corporation with the donation of the Titan V GPU.

References

1. Pratap, A.: Analysis of big data technology and its challenges. Int. Res. J. Eng. Technol. (IRJET) **6**, 5094–5098 (2019)
2. Zeghida, D., Meslati, D., Bounour, N.: Bio-IR-M: a multi-paradigm modelling for bio-inspired multi-agent systems. Informatica **42**(3) (2018)
3. Wooldridge, M., Jennings, N.R.: Intelligent agents:theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
4. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Auton. Agent. Multi-Agent Syst. **14**(1), 5–30 (2007). https://doi.org/10.1007/s10458-006-0012-0
5. Sun, R., et al.: Cognition and Multi-agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge University Press, Cambridge (2006)
6. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. Knowl. Eng. Rev. **19**(4), 281–316 (2004)
7. Jean-Pierre, M., Christine, B., Gabriel, L., Pierre, G.: Bio-inspired mechanisms for artificial self-organised systems. Informatica **30**(1) (2006)
8. Fewell, J.H., Harrison, J.F.: Scaling of work and energy use in social insect colonies. Behav. Ecol. Sociobiol. **70**(7), 1047–1061 (2016). https://doi.org/10.1007/s00265-016-2097-z
9. Ward, A.J., Herbert-Read, J.E., Sumpter, D.J., Krause, J.: Fast and accurate decisions through collective vigilance in fish shoals. Proc. Natl. Acad. Sci. **108**(6), 2312–2315 (2011)
10. Muro, C., Escobedo, R., Spector, L., Coppinger, R.: Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations. Behav. Process. **88**(3), 192–197 (2011)

11. Jøsang, A.: Subjective Logic. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-42337-1
12. He, C., Li, Y.: A survey of intelligent decision support system. In: 2017 7th International Conference on Applied Science, Engineering and Technology (ICASET 2017), pp. 201–206. Atlantis Press (2017)
13. Garro, A., Mühlhäuser, M., Tundis, A., Mariani, S., Omicini, A., Vizzari, G.: Intelligent agents and environment. In: Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics, p. 309 (2018)
14. Railsback, S.F., Grimm, V.: Agent-Based and Individual-based Modeling: A Practical Introduction. Princeton University Press, Princeton (2019)
15. Michel, F., Ferber, J., Drogoul, A.: Multi-agent systems and simulation: a survey from the agent community's perspective. In: Multi-Agent Systems, pp. 17–66. CRC Press (2018)
16. Pipattanasomporn, M., Feroze, H., Rahman, S.: Multi-agent systems in a distributed smart grid: design and implementation. In: 2009 IEEE/PES Power Systems Conference and Exposition. PSCE2009, pp. 1–8. IEEE (2009)
17. Fernández-Isabel, A., Fuentes-Fernández, R.: An agent-based platform for traffic simulation. In: Corchado, E., Snášel, V., Sedano, J., Hassanien, A.E., Calvo, J.L., Ślęzak, D. (eds.) Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011. Advances in Intelligent and Soft Computing, vol. 87, pp. 505–514. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19644-7_53
18. González-Briones, A., De La Prieta, F., Mohamad, M.S., Omatu, S., Corchado, J.M.: Multi-agent systems applications in energy optimization problems: a state-of-the-art review. Energies **11**(8), 1928 (2018)
19. Conati, C., Klawe, M.: Socially intelligent agents in educational games. In: Dautenhahn, K., Bond, A., Cañamero, L., Edmonds, B. (eds.) Socially Intelligent Agents. Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 3, pp. 213–220. Springer, Boston (2002). https://doi.org/10.1007/0-306-47373-9_26
20. Fernández-Isabel, A., Fuentes-Fernández, R., de Diego, I.M.: Modeling multi-agent systems to simulate sensor-based smart roads. Simul. Model. Pract. Theory **99**, 101994 (2020)
21. Salehinejad, H., Talebi, S.: Dynamic fuzzy logic-ant colony system-based route selection system. Appl. Comput. Intell. Soft Comput. **2010**, 13 (2010)
22. Yan, X., Li, L.: Ant agent-based QoS multicast routing in networks with imprecise state information. In: Shi, Z.-Z., Sadananda, R. (eds.) PRIMA 2006. LNCS (LNAI), vol. 4088, pp. 374–385. Springer, Heidelberg (2006). https://doi.org/10.1007/11802372_36
23. Bellifemine, F., Poggi, A., Rimassa, G.: JADE: a FIPA2000 compliant agent development environment. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 216–217 (2001)
24. Yang, Y.J., Sung, T.-W., Wu, C., Chen, H.-Y.: An agent-based workflow system for enterprise based on FIPA-OS framework. Expert Syst. Appl. **37**(1), 393–400 (2010)
25. Spade: scheduler for parallel and distributed execution from mobile devices
26. Crooks, A.T., Heppenstall, A.J.: Introduction to agent-based modelling. In: Heppenstall, A., Crooks, A., See, L., Batty, M. (eds.) Agent-Based Models of Geographical Systems, pp. 85–105. Springer, Dordrecht (2012). https://doi.org/10.1007/978-90-481-8927-4_5

27. Shehory, O., Sturm, A. (eds.): Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54432-3
28. Pavón, J., Gómez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: Agent-Oriented Methodologies, no. 9, pp. 236–276 (2005)
29. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Auton. Agent. Multi-Agent Syst. **8**(3), 203–236 (2004). https://doi.org/10.1023/B:AGNT.0000018806.20944.ef
30. Goulson, D.: Bumblebees: Behaviour, Ecology, and Conservation. Oxford University Press on Demand, Oxford (2010)
31. Korb, J., Thorne, B.: Sociality in termites. In: Comparative Social Evolution, pp. 124–153 (2017)
32. MacDonald, J., Deyrup, M.: The social wasps (hymenoptera: Vespidae) of Indiana. Great Lakes Entomol. **22**(3), 7 (2017)
33. Seid, M.A., Traniello, J.F.: Age-related repertoire expansion and division of labor in pheidole dentata (hymenoptera: Formicidae): a new perspective on temporal polyethism and behavioral plasticity in ants. Behav. Ecol. Sociobiol. **60**(5), 631–644 (2006). https://doi.org/10.1007/s00265-006-0207-z
34. Jandt, J.M., Dornhaus, A.: Spatial organization and division of labour in the bumblebee Bombus impatiens. Anim. Behav. **77**(3), 641–651 (2009)
35. Tschinkel, W.R.: The nest architecture of the florida harvester ant, pogonomyrmex badius. J. Insect Sci. **4**(1), 21 (2004)
36. Cicirello, V.A., Smith, S.F.: Wasp nests for self-configurable factories. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 473–480 (2001)
37. de Oliveira, V.M., Campos, P.R.: The emergence of division of labor in a structured response threshold model. Phys. A: Stat. Mech. Appl. **517**, 153–162 (2019)
38. Duarte, A., Pen, I., Keller, L., Weissing, F.J.: Evolution of self-organized division of labor in a response threshold model. Behav. Ecol. Sociobiol. **66**(6), 947–957 (2012). https://doi.org/10.1007/s00265-012-1343-2
39. Shafer, G.: A Mathematical Theory of Evidence, vol. 42. Princeton University Press, Princeton (1976)
40. Zadeh, L.A.: Fuzzy logic. Computer **21**(4), 83–93 (1988)
41. Sidhu, A.S.: Recommendation framework based on subjective logic in decision support systems, Ph.D. thesis, University of Windsor (2014)
42. Jøsang, A., Hayward, Pope, S.: Trust network analysis with subjective logic. In: Proceedings of the 29th Australasian Computer Science Conference (ACSW 2006), pp. 885–894. Australian Computer Society (2006)
43. Liu, Y., Li, K., Jin, Y., Zhang, Y., Qu, W.: A novel reputation computation model based on subjective logic for mobile ad hoc networks. Future Gener. Comput. Syst. **27**(5), 547–554 (2011)
44. Jøsang, A., Bondi, V.A.: Legal reasoning with subjective logic. Artif. Intell. Law **8**(4), 289–315 (2000). https://doi.org/10.1023/A:1011219731903
45. Khan, M.W., Wang, J., Ma, M., Xiong, L., Li, P., Wu, F.: Optimal energy management and control aspects of distributed microgrid using multi-agent systems. Sustain. Cities Soc. **44**, 855–870 (2019)
46. Gerbet, T., Kumar, A., Lauradoux, C.: A privacy analysis of Google and Yandex safe browsing. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 347–358. IEEE (2016)

47. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254 (2009)
48. McGrath, D.K., Gupta, M.: Behind phishing: an examination of phisher modi operandi. In: LEET, no. 4, p. 8 (2008)