International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume 15 Issue 01 Year 2023

Intelligent Agents Model with JADE for Scheduling Analysis and Correction of Real-Time Systems

Walid Karamti
Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia
Data Engineering and Semantics Research Unit, Faculty of Sciences of Sfax, University of Sfax, Sfax 3052, Tunisia
Imad Al-Sgir
Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia
w.karamti@qu.edu.sa

Article History

Received: 10 March 2023
Revised: 18 April 2023
Accepted: 19 May 2023

Abstract

This research proposes a new model for analyzing and correcting non-schedulable partitions in real-time multiprocessor systems, specifically in the context of fault tolerance in distributed networks. The need for such a model arises from current techniques for correcting non-schedulable partitions that must be revised and repartitioning all tasks across processors. The proposed model is based on intelligent agents and implemented using the JADE platform. The model consists of (1) a supervisor agent in the first layer that distributes tasks and manages system correction when a non-schedulable partition is detected; and (2) a second layer composed of partition agents that analyze schedulability, request corrections, and negotiate with the supervisor for additional tasks to correct the entire system. The effectiveness of the proposed model is demonstrated through a case study. Quantitative analysis shows that the proposed model improves fault tolerance in distributed systems and has the potential for further enhancement by adding communicative tasks, heterogeneous processors, and other improvements.

Keywords: Real-Time System, Scheduling Analysis and Correction, Fault Tolerance, Multi-Agent System, JADE

1. Introduction

With the rapid development of technology and the multiplicity of smart devices and applications, it has become necessary for most of them to be compatible with Real-Time Systems (RTS). RTS has been deployed in fields such as industrial, health, automobiles, avionics, etc.

RTS has a significant constraint, and breaching this constraint leads to failure. This constraint is known as the deadline. Any task must be completed before its deadline [31]. Before deploying the RTRT application on the architecture, the designer must ensure that the system will be scheduled without issues and that all tasks will meet their corresponding deadlines. To do that, it is crucial to analyze the system and correct all the failures early. The analysis depends on two major factors: the application type and the architecture used. We distinguish two types of architecture. First, single-processor architecture is the most widely used in the literature [19], [15], where optimal scheduling

algorithms exist for all possible types of system applications. Second, multiprocessor architecture is based on the distribution of the application tasks over the available processors. This problem, known as an NP-Hard [16], has caught the attention of researchers over the past two decades, and the results continue to improve [17], [11]. There are two types of multiprocessor scheduling. The first is global scheduling, which allows all jobs to continue running on any available processor. However, it is crucial to consider the cost of migration and preemption [4], [22].

Lowering the costs of preemption and migration is at the core of the second type of multiprocessor scheduling, namely the partitioned family [30]. In fact, during the assignation stage, the tasks are dispersed among the processors. Each partition is treated as a single-processor scheduling issue to take advantage of the current optimal scheduling policy [20]. To guarantee the feasibility of the RTS, a scheduling analysis of all partitions is then performed.

The scheduling analysis is approached in the literature using various strategies to safeguard the RTS from errors. The formal method, in particular, presents the most secure scheduling analysis method.

In practice, the system's characteristics and the attributes that need to be verified determine which formal technique is most appropriate. For scheduling analysis, we distinguish between two primary approaches: analytical and model-checking techniques [21].

Analytical verification offers algorithms with polynomial complexity for task scheduling verification of a group of tasks, in contrast to model-checking approaches [24]. Compared to other procedures, the analysis's results are delivered quite soon. This explains why this approach is so prevalent in practice.

Nevertheless, despite the wealth of research conducted in this area, the suggested schedule partitioning methods frequently have different issues. The probability of non-schedulability is most frequently increased by the failure to provide a suitable partitioning solution from the first assignment iteration. In reality, the partitioner is requested for a second regeneration from an exponential number of viable partitioning solutions to fix the schedulability. Correcting non-schedulable partitions is, in fact, an expensive process.

To lower the rising expense of regeneration, attempting to correct the solution under consideration is compelling. As a classical solution for reduced problem complexity, the designer relies on their knowledge to re-assign some tasks to correct the schedulability. However, a complex system presents the challenge of producing a new autonomous scheduling analysis method that can conduct analysis and decide to correct the schedulability locally. Should all the correction attempts fail, the partitioner would be contacted to regenerate a new partitioning.

The Multi-Agent Systems (MAS) paradigm is among the most relevant approaches toward analyzing a system's behaviours and making decisions to achieve a specific goal. Thus, as our main objective in this paper, we use a MAS model for the scheduling analysis and correction of RTS.

The main contribution of this study is introducing a multi-agent system (MAS) model that employs intelligent, cooperative agents to analyze the schedulability of various partitions and rectify any non-schedulable partitions. The Contract Net protocol is used for agent communication, and the correction process involves calculating the CPU's utilization bound.

The rest of the manuscript is organized as follows: Section 2 presents a literature review of available MAS platforms. Next, the suggested MAS model for RTS scheduling analysis and correction is given in section 3. The focus of this paper is on periodic independent tasks. After that, section 4 provides an in-depth presentation of the case study, the implementation requirements, and the ability of the suggested technique to correct all the mentioned non-schedulable partitions. Section 5 presents the conclusion and provides suggestions for future research work.

2. Literature Review

2.1 Multi-Agent Systems

A multi-agent system (MAS) is a group of autonomous agents interacting with each other and their surroundings [33], [2]. An agent is a self-contained, intentional software entity that can communicate with other agents through an information exchange such as messages. Agents work together to achieve a specific goal by performing activities. They can decide collaboratively on the most appropriate aim and then perform tasks related to their shared goal.

The MAS comprises an environment, various objects and agents, their relationships, and a set of actions these components can perform. The set of actions must be able to alter the environment and exert control over the rest of the components. These systems are utilized in a variety of fields, notably simulations. They offer the ability to create artificial environments to test theories about specific behaviours (e.g. Real-Time Systems modelling and simulation [8]).

Agent-oriented systems can be developed using a variety of approaches and methodologies [12]. These contain a set of strategies and recommendations for developing such systems in a systematic and coordinated manner. Several agent-oriented systems are Model Driven Engineering (MDE) based and come with toolkits.

According to their focus, these approaches can be divided into three categories [14]. The first category comprises multi-agent system methodologies, which focus on each agent individually, considering the point of view of the collective system. The agent's role and its impacts on the system are discussed in [29].

The second category comprises requirement-driven methodologies, which focus on eliciting requirements using structured development methods appropriate to those used in the programming paradigm [7].

The third category comprises agent-oriented methodologies, representing the difficulties that need to be solved in the systems. These difficulties are resolved by extending notions from an object-oriented approach and applying them to the context of Multi-Agents Systems [28], [34].

However, our primary goal is to analyze the behaviour of task partitions. We do not use the MAS for scheduling. Thus, we will draw from the results of existing work to present the different platforms, interaction methods, and communication protocols used.

Kravari and Bassiliades proposed a survey paper in [23]. This work presents a list of 24 agents' platforms, most importantly AGLOBE [32], AnyLogic [5], Cougaar [18], CybelePro [26], Repast[27], Jade [3] and Jadex [6].

Hence, we present in Table. 1 A comparison of the platforms based on the programming languages, communication network, standard protocols compatibility, adopted interaction method, agent architecture, cost, and popularity.

2.2 MAS and RTS Scheduling Analysis

In recent years, several works have addressed the problem of RTS scheduling. The existing works have focused on modelling scheduling algorithms with MAS models. Each running task is presented with an agent, and the cooperation between the agents must respect the modelled algorithm. However, these works mainly focus on local scheduling, and the developed platforms are tested through different RTS applications. For instance, in [9], the authors presented a platform that supports scheduling algorithms such as RMRM, FCFS, and EDF.

Similarly, in [13], the authors used a MAS simulator called Marzhin to simulate scheduling a set of real-time tasks executed on a single given processor. While these works address scheduling analysis, they operate at a lower level of abstraction than scheduling analysis and correction. In this article, we propose a new scheduling analysis and correction model based on the MAS model. The aim is to benefit from the behaviour of autonomous agents and the cooperation between agents for possible system corrections. To the best of our knowledge, no existing MAS model was developed for RTS scheduling analysis and correction. While recent work has been presented in [25] to propose a possible RTS scheduling analysis and correction model, it was only a theoretical description of the model. Many components were discarded to describe the agents adequately. Hence, we are motivated in this article to develop and implement a new model with all the necessary components using the JADE platform. While existing works have some advantages in modelling scheduling algorithms, they can only perform system corrections with costly repartitioning. Our proposed approach aims to overcome this limitation and improve the scheduling analysis and correction process for RTS.

3. Intelligent Agents Model for RTS Scheduling Analysis and Correction

This section presents the proposed MAS model for the partitioned multiprocessor RTS scheduling analysis and correction. We begin by presenting the model architecture, which identifies all of the model's components. Next, we provide a specification of the environment and the

intelligent agents used. Specifically, we detail the various types of agents and their corresponding behaviours. Finally, we describe how the agents communicate with each other and their actions on the environment to correct non-schedulable partitions.

Table 1. Platforms Comparison

Platform	**Programming**	**Commun.**	**Inter.**	**Agents**	**Cost**		**Popularity**
	Language	**network**	**method**	**arch.**			
		And proto-					
		Cols stand.					
		compatibility					
AGLOBE [32]	Java	Partially FIPA (ACL), GIS	ACL	hybrid	Free		Medium
AnyLogic [5]	Java	GIS, 3D capa-abilities	Messages	Cognitive	AnyLogic advanced	Ad-$6,199	Medium
	UML-RTRT (UML for real-time)				Professional $15,800, University Researcher License $3,500, Educa-National Licenses $485		
Cougar [18]	Java	unspecified	Cougar Message Transport Service (MTS)	Cognitive	Free		Low
CybelePro [26]	Java	unspecified	Messages	Cognitive	Commercial $1000-$4000, Academic $600-$2400		Low
Repast [27]	Java, C++	FIPA	Messages	Free	Medium		
Jade [3]	Java	FIPA, CORBA	Messages	Reactive, Cogniti ve, Hybrid	Free		High (most popular)
Jadex [6]	Java (plus use of XML)	FIPA	Messages	Cognitive (BDI)	Free		High

3.1 Proposed Model

Figure 1 depicts the proposed intelligent agent model for the scheduling analysis and correction of partitioned real-time systems using the JADE platform. The model is composed of two main parts. The first part is the environment, which defines the real-time system under consideration, including task characteristics and the various partitions. The second part is the agent network, which defines the different intelligent agents, their behaviours, and the communication protocol utilized. In the following sections, we describe each component of the model in greater detail.

3.2 Environment

A partitioned multiprocessor RTS can be defined using the following quadruplets:

$$RTS = \langle Task, Proc, Alloc, Prec \rangle \quad (1)$$

With:

• Task: defines a set of n tasks {T1, ... ,Tn}, where each task, Ti ∈ Task, is defined with the set $Ti = \langle Ri, Pi, Di, Ci \rangle$:

Ri: Ti's first arrival time,

Pi: Ti's activation period,

Di: Ti's deadline.

Ci: Ti's worst execution time.

• Proc: a set of m identical processors: {Proc1, ..., Procm}

• Alloc: defines the n tasks' distribution over the m processors.

$$Alloc: Task \times Proc \rightarrow \{0; 1\} \quad (2)$$

$$(T_i; Proc_j) \rightarrow \begin{cases} 0: \text{The task } T_i \text{ is not allocated to the proc } Proc_j \\ 1: \text{The task } T_i \text{ is allocated to the proc } Proc_j \end{cases}$$

• Prec: defines the tasks' precedence relationships.

$$Proc: Task \times Task \rightarrow \{0; 1\} \quad (3)$$

$$(T_i; T_j) \rightarrow \begin{cases} 0: \text{The task } T_i \text{ does not precede the task } T_j \\ 1: \text{The task } T_i \text{ precede } T_j \end{cases}$$

All the necessary data concerning the tasks, the allocation, and the precedence are recovered from a partitioner tool as input.

We have used the JADE platform to implement the RTS as an environment for the proposed agent model. Indeed, to specify the environment with JADE, we proposed a Java class called input describing the RTS, including the tasks' specifications, the CPUs, the tasks' precedence, and allocation descriptions.

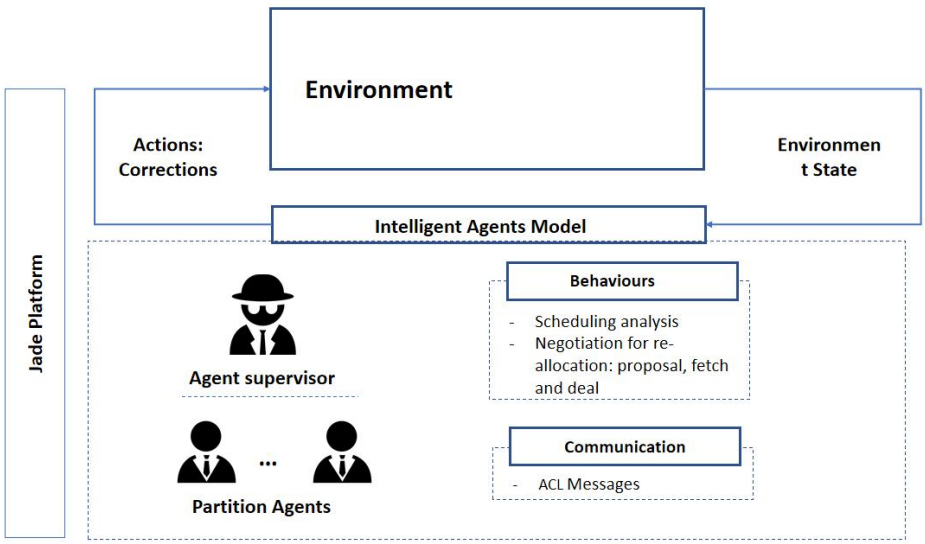

Figure 1. Proposed Intelligent Agents Model for RTS Scheduling Analysis and Correction

3.3 Intelligent Agent Specification

In the proposed model, we have relied on the multi-agent system to achieve this system's advantages and benefit from its qualitative services. As shown in Figure 1, we proposed two types of agents: agent supervisor and partition agent.

For an RTS, one Agent supervisor and m partition agents will be created. The setting operation of the agents is conducted based on the specified environment. Based on the Alloc description, the supervisor agent distributes the tasks over the partition agents to configure their settings.

Once the agent partition setting is completed, scheduling analysis behaviour is initiated to check the processor's ability and receive the allocated tasks' load.

At the end of the analysis, the agent notifies the supervisor if a non-schedulable task is detected by sending the needed load to release to correct the schedulability. Otherwise, no notification is required, and the agent keeps the information about the available processor load in case a proposal from the supervisor is received for a possible new task allocation.

When the supervisor receives a request, a proposal is sent to the partition agents who have no requests to check their ability to receive a new task's load.

Each partition agent answers the supervisor's request after a new scheduling analysis with an acceptance or a rejection. After that, the supervisor makes a deal with the agent representing the best fit for the requested load, and then a repartitioning action is performed on the environment.

To summarize, all requests are answered so that the RTS schedulability is corrected and it is safe to deploy the application on the architecture. Otherwise, the system is declared non-schedulable, and a new design iteration is recommended.

In the following section, we define the behaviour of the leading agents.

3.3.1 Partition Agent Behaviours

Once the partition queue is initialized from the supervisor agent, each partition agent defines four behaviours to start: scheduling analysis behaviour, correction request behaviour, receiving proposal behaviour, and sending response behaviour.

- Scheduling analysis behaviour: This behaviour can analyze the schedulability of the allocated tasks on the CPU. This behaviour runs during the creation of the initial partition or to check the capacity to receive an additional task.

The proposed agent can analyze the schedulability of independent periodic tasks. This analysis is based on the equation defined by Liu and Layland [24] (eq. 5). The analysis is based on the CPU's ability to hold the workloads of all its partitions. This problem is known as a bin-packing [10] problem, and the proposed equation (eq. 5) is proved optimal for the periodic independent tasks. Therefore, each partition (eq. 4) is analyzed as a single processor.

Let $\mathrm{Proc_j} \in \mathrm{Proc}$, the tasks' partition of $\mathrm{Proc_j}$ called $\mathrm{Part(Proc_j)}$ is defined as follows:

$$\mathrm{Part(Proc_j)} = \{\forall \mathrm{Ti} \in \mathrm{Task};\ \mathrm{Alloc(Ti; Proc_j)} = 1\} \quad (4)$$

$$\mathrm{U(Proc_j)} = \sum_{\mathrm{T_i} \in \mathrm{Part(Proc_j)}} \frac{\mathrm{C_i}}{\mathrm{P_i}} \leq 1 \quad (5)$$

Three possible results can be reached from the analysis:

$\mathrm{U\ (Proc_j)} < 1$: The partition is schedulable and able to receive more tasks with a load up to $(1 - (\mathrm{U(Proc_j)})$.

$\mathrm{U\ (Proc_j)} = 1$: The partition is schedulable with no space to receive additional tasks.

$\mathrm{U\ (Proc_j)} > 1$: The partition is non-schedulable. Otherwise, it can be fixed if the exceeded load $(\mathrm{U\ (Proc_j)} - 1)$ is released and taken by another partition. This load can be specified with one task or a set of tasks. Thus, a request for correction is required, and the second behaviour, the Correction request behaviour, is then called to run.

This behaviour is called for each proposal from the supervisor agent for possible addition of tasks to proceed with a system correction.

- Correction request behaviour: This behaviour is called if a non-schedulable partition is detected. This behaviour is composed of two steps. First, the partition task queue is sorted from highest to lowest load. Second, the candidate tasks are selected for release to correct the schedulability. Based on multiple extensive experiments, we are convinced that fewer load tasks are more likely to be accepted for scheduling on other partitions. Thus, the current behaviour defines the set of tasks to be released and then requested by the supervisor agent to find available space on the other schedulable partitions.

Sending the request causes the tasks to be removed from the partition's queue and added to the supervisor's queue.

• Receiving proposal behaviour: According to the Foundation of Intelligent Physical Agents (FIPA) standard [1], an agent is the primary actor in an area. It can combine various service capabilities to produce an integrated and unified operational model, including access to external software, human users, and communications networks.

The FIPA standards cover Agent Communication Language (ACL) messages, message exchange interaction protocols, speech act theory-based communicative actions, and content language representations. JADE includes the FIPA protocol, and the communications are designed with the ACL. The object produced from the agent transparency class is used to implement inter-agent communication. It chooses the optimum path and balances message semantics and format to facilitate their exchanges. All messages follow the ACL requirements, including:

(1) Message source and communicative intention ("performativity");

(2) Message content, language, conversation IDID, and ontology.

The receiving proposal behaviour consists of receiving the supervisor agent's proposal to receive an additional task. Its main instructions are:

(1) Call for scheduling analysis behaviour to verify its capability to receive the additional task;

(2) Prepare the response message to the supervisor agent of accepting or declining the proposal. If it accepts, it must also send the remaining remaining space. Thus, the sending decision behaviour is called.

• Sending decision behaviour: This behaviour aims to send the prepared message to the supervisor agent to inform it of the possible state of the partition if the requested task is added to the queue of the partition.

3.3.2 Supervisor Agent Behaviours

The supervisor agent is initially responsible for distributing the tasks to the different partitions and then receiving requests from non-schedulable agent partitions. After that, the supervisor agent sends proposals to other agents, negotiates and makes decisions for the scheduling correction. To do this, two main behaviours are proposed:

• Receiving request and sending proposal behaviour: This behaviour starts once a correction request behaviour from a partitioning agent is executed. A message is received containing the tasks requiring a new reassignment. The first action is the update the supervisor's queue by adding the received tasks. After that, the supervisor prepares a message describing a proposal of partition agents corresponding to the task at the head of the queue. Finally, a request is sent to the destinations. This behaviour is repeated until all tasks in the queue have been processed.

• Negotiation behaviour: The role of this behaviour is to receive the decisions of the agent partitions. Once all the decisions have been collected, the agent sorts all the remaining available spaces received from least to highest and selects the highest. In fact, because of this, the agent ensures the selection based on the most suitable algorithm, thereby improving the possibility of re-assigning more tasks in future proposals. This action presents the best-fit implementation to assign the tasks and maintain the highest remaining space to receive additional tasks.

After that, the agent sends a confirmation message to the corresponding partition agent and updates its queue before moving on to the next task.

4. Case Study

To evaluate the effectiveness of the proposed model, we consider the same RTS described in [25] to show that our model can obtain the same theoretical results. First, we present the description of the RTS environment. Next, we describe how our model is executed to analyze the schedulability. Finally, we present how the communications between the agents are established to cooperate and correct the system.

4.1 Environment Description

In Table 2, we consider fifteen independent tasks with a deadline on request ($P_i = D_i$) and a simultaneous start ($R_i = 0$) .We consider four identical processors. The initial partitioning is described in Table 3.

Table 2. Tasks Description

$Task_{id}$	P_i	C_i	$Task_{id}$	P_i	C_i	$Task_{id}$	P_i	C_i
0	16	3	5	9	4	10	7	2
1	3	1	6	8	3	11	7	1
2	8	1	7	7	3	12	15	3
3	15	8	8	16	2	13	9	1
4	7	2	9	9	2	14	15	1

Table 3. Initial Partitioning

P1	P2	P3	P4
T0	T1	T3	T4
T2	T5	T12	T7
T6	T9	T14	T10
T8	T13		T11

4.2 Model Initialization

Based on the initial partitioning presented in Table 3, a supervisor agent is created, thereby allowing four agent processor partitions to be created. The supervisor agent then assigns the tasks to their corresponding processors. The results of this step are presented in Table 4- step 1. At this stage, the model is initialized, and the scheduling analysis behaviour can be initiated to assess the system's feasibility.

4.3 Based Model Execution

The partition agents initiate the scheduling analysis process by executing the scheduling analysis behaviour, which aims to determine the utilization bound of the processor it is assigned to. The calculation is derived from equation 5. We present the results in Table 4- step 2 to demonstrate the effectiveness of our experiments' real-time system (RTS).

Upon conducting the initial partitioning scheduling analysis, it was found that processors P2 and P4 were overloaded, while P1 and P3 were deemed schedulable. Consequently, the non-schedulable tasks were communicated to the supervisor for reassignment, allowing P2 and P4 to alleviate their workloads. Therefore, the correction request behaviour is run.

The initial step of the correction behaviour involves selecting a set of tasks for reassignment. In the case of P2 and P4, the tasks T13 and T11 with the lowest workload are sufficient to alleviate the processors. Consequently, a request is sent to the supervisor to inquire about possible corrections (Table 4, steps 3-4).

Table 4. Output Description- Correction of T11

Steps	Actions
1	Supervisor distributes the tasks to the processor agents: P1: T0, T2, T6, T8 P2: T1, T5, T9, T13 P3: T3, T12, T14 P4: T4, T7, T10, T11
2	Scheduling Analysis results: P1: schedulable, U= 0.8125 P2: non-schedulable, U=1.1111 P3: schedulable, U=0.8 P4: schedulable, U=1.1429
3	Agent Processor P2 request a correction action T13 is selected to be re-allocated Agent Processor P2 sends T13 to the Supervisor Agent
4	Agent Processor P4 request a correction action T11 is selected to be re-allocated Agent Processor P4 sends T11 to the Supervisor Agent
5	Supervisor Agent receives request from the Agent Processor P2 Supervisor Agent receives request from the Agent Processor P4
6	Supervisor Agent sends T11 to the Processor Agent P1 Supervisor Agent sends T11 to the Processor Agent P3
7	Agent Processor P1 do a Scheduling Analysis Agent Processor P3 do a Scheduling Analysis
8	Agent Processor P1 sends a proposal to the Supervisor Agent with U=0.9554 Agent Processor P3 sends a proposal to the Supervisor Agent with U=0.9429
9	Supervisor Agent analyze the received proposal Supervisor Agent assign T11 to the Agent Processor P1 P1: schedulable, U= 0.9554 P2: non-schedulable, U=1.0 P3: schedulable, U=0.8 P4: schedulable, U=1.0

Once the supervisor agent receives a request, the corresponding task is added to the supervisor′ s queue. Starting at the top of the queue, the supervisor generates a proposal for the partition agents, who are presented with the schedulable processors that could potentially receive the task. In our scenario, the supervisor agent sends the tasks T11 to P1 and P3 (Table 4, steps 5-6).

Subsequently, the partition agents perform a scheduling analysis for each received proposal before formulating a response to the supervisor. Once the examination is completed, the partitioning agent sends a confirmation to the supervisor, providing the updated utilization bound value. In the current scenario, the partition agents assigned to processors P1 and P3 submit to the supervisor their proposals containing the utilization value if it accepts the requested task, which is U=0.9554 and U=0.9429, respectively. The results are depicted in Table 4, steps 7-8.

The final step entails selecting the best proposal among the responses from the partition agents and negotiating the optimal offer based on the best-fit algorithm. Specifically, the proposal with the least available space remaining is chosen. Consequently, this algorithm improves the probability of assigning subsequent non-schedulable tasks in the supervisor queue to other processors. Finally, the tasks are assigned to their corresponding partition. In this case, task T11 is assigned to processor P1 and removed from the supervisor queue (Table 4, step 9).

After finding a new assignment for task T11, the supervisor agent selects task T13, and the process is repeated. In Table 5, the supervisor sends two requests to the partition agents, P1 and P3. However, only one proposal from P3 is received because P1 cannot sche2dule the task. Thus, the supervisor concludes the contract with P1.

Once the supervisor queue has cleared, the system can be declared schedulable. However, if a request is made without a proposal, the correction action fails, and a new repartitioning is necessary.

Table 5. Output Description- Correction of T13

Steps	Actions
10	Supervisor Agent sends T13 to the Processor Agent P1 Supervisor Agent sends T13 to the Processor Agent P3
11	Agent Processor P1 do a Scheduling Analysis Agent Processor P3 do a Scheduling Analysis
12	Agent Processor P3 sends a proposal to the Supervisor Agent with U=0.9111
13	Supervisor Agent analyze the received proposal Supervisor Agent assign T13 to the Agent Processor P3 P1: schedulable, U= 0.9554 P2: non-schedulable, U=1.0 P3: schedulable, U=0.911 P4: schedulable, U=1.0

4.4 Discussion

The proposed model for the case study has demonstrated its ability to correct the initial partitioning without requiring a new repartitioning. This represents a significant improvement over existing approaches, as our MAS model is the first to provide RTS scheduling analysis and correction. While [25] has proposed a theoretical model, it needs more critical details. In our work, we have presented a detailed model that specifies intelligent agents. Specifically, we have described the agents' architecture, communication protocols, behaviours and objectives. We have also utilized the Contract Net protocol and implemented our model on the JADE platform. During the validation process, we tested the model using various types of RTS. Our results revealed that the model's success depends on the initial partitioning. For balanced partitions, the model can accurately identify corrective actions. However, for non-balanced partitions and high processor utilization rates, the model needs help identifying a schedulable solution. This limitation stems from our model's inability to consider a task as a set of frames and to recommend frame reallocation for system correction. Thus, we recommend frame-based corrections to improve our model's ability to handle dependent tasks. At present, saturated systems remains a significant challenge.

5. Conclusions

The scheduling analysis and correction of real-time systems (RTS) at the process level is a complex and challenging research area with continued development. One of the most commonly used methods for correcting non-schedulable partitions involves repartitioning tasks among processors. However, this method is known to be an NP-hard problem, making it a costly solution. To address this issue, proposing alternative solutions for local corrections on non-schedulable partitions is a promising approach for reducing the complexity of the process.

This paper proposes a correction approach based on a multi-agent system (MAS) model consisting of a two-layer model with intelligent, cooperative agents for independent periodic RTS. The model was implemented using JADE, and we conducted experiments to test its correctness and limitations. Our results demonstrate that the proposed MAS model efficiently analyzes and corrects non-schedulable partitions without costly repartitioning. The Contract Net protocol facilitates effective communication between the agents, while the utilization bound calculation accurately identifies and corrects non-schedulable partitions.

However, the success of the corrections depends on the quality of the initial partitioning. Our proposed model achieves its objective if the initial solution is constructed using a balanced

partitioning algorithm, such as a bin-packing algorithm. However, if the initial partitioning is not well-balanced, the model may fail, even if potential correction scenarios exist. Furthermore, the proposed model is restricted to a specific type of RTS, independent periodic tasks. Therefore, we aim to enhance our model by including dependent tasks, improving our model's ability to correct a broader range of diverse and complex systems.

Overall, our experiments demonstrate the effectiveness of the proposed MAS model in correcting non-schedulable partitions in independent periodic RTS. The outcomes of our research provide a promising approach for future studies to develop models that can address a broader range of RTS complexities.

6. Acknowledgment

The authors gratefully acknowledge Qassim University, represented by the Deanship of Scientific Research, for the financial support for this research under the number (COC-2022-1-1-J-26037) during the academic year 1444 AH / 2022 ADAD.

References

[1] FIPA, "Interaction Protocol specifications," Last access, December 2022.

[2] R. Alhanani, J. Abouchabaka, and R. Najat., "Cdsmip: Cds-based multiple itineraries planning for mobile agents in a wireless sensor network," *International Journal of Communication Networks and Information Security*, vol. 11, 2019, pp. 202–211.

[3] F. Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade: a white paper. vol. 3, 2003, pp. 6–19.

[4] M. Bertogna and S. Baruah. "Tests for global edf schedulability analysis," *Journal of Systems Architecture*, Special Issue on Multiprocessor Real-time Scheduling, vol. 57, no. 5, pp. 487–497, 2011.

[5] A. Borshchev. "The big book of simulation modelling: multimethod modelling with AnyLogic 6," *AnyLogic North America*, Lisle, IL, 2013.

[6] L. Braubach and A. Pokahr, "The Jadex Project: Simulation," Berlin, German: Springer Berlin Heidelberg, 2013, pp. 107–128.

[7] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos,"Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, no. 8, 2004, pp. 203–236.

[8] D. Calvaresi, Y-D. Cid, M. Marinoni, A-F. Dragoni, A. Najjar, and M. Schumacher. "Real-time multi-agent systems: rationality, formal model, and empirical results," *Auton, Agents Multi-Agent Syst.*, vol.35, no.1, 2021, pp.12.

[9] D. Calvaresi, M. Marinoni, L. Lustrissimini, K. Appoggetti, P. Sernani, A. F. Dragoni, M. Schumacher, and G. C. Buttazzo. "Local scheduling in multi-agent systems: Getting ready for safety-critical scenarios," In Francesco Belardinelli and Estefania Argente, editors, Multi-Agent Systems and Agreement Technologies -15th European Conference, EUMAS 2017, and 5th International Conference, AT 2017, Évry, France, December 14-15, 2017, Revised Selected Papers, vol.10767, Lecture Notes in Computer Science, Springer, 2017, pp. 96–111.

[10] E. Coffman and J. Csirik, "Performance Guarantees for One-Dimensional Bin Packing," 2007, pp. 32–1.

[11] E-G. Coffman and C. János., "Performance guarantees for one-dimensional bin packing," In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007.

[12] H. Dam and M. Winikoff. "Comparing agent-oriented methodologies," Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2003, pp. 3030.

[13] P. Dissaux, O. Marc, S. Rubini, C. P. Kankeu Fotsing, V. Gaudel, F. Singhoff, A. Plantec, V. Nguyen-Hong, and H. Nam Tran. "The intelligent project: Multi-agent scheduling simulation of real-time architectures," In *Proceedings of the 7th European Congress ERTSS Embedded Real Time Software and System*, Toulouse, France, Feb., 2014.

[14] A. Fernández-Isabel and R. Fuentes-Fernández. "Extending a generic traffic model to specific agent platform requirements," *Computer Science and Information Systems*, vol. 14, no. 1-1, 2017.

[15]D. D. Gajski, F. Vahid, and S. Narayan., "A system-design methodology: executable-specification refinement," In *Proceedings of European Design and Test Conference EDAC-ETC-EUROASIC*, 1994, pp. 458–463.

[16]M. R. Garey and D. S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness," USA:W. H. Freeman, 1990.

[17]J. Goossens and P. Richard, "Multiprocessor Real-Time Scheduling," Berlin, German: Springer, pp. 1–33, 2017.

[18]A. Helsinger and T. Wright., "Cougar: A robust configurable multi-agent platform," 2005, pp. 1–10.

[19]J., L. Hennessy and D. A. Patterson, "Computer Organization and Design (2nd Ed.): The Hardware/Software Interface," San Francisco, USA: Morgan Kaufmann Publishers Inc., 1997.

[20]K. S. Hong and J. Y. T. Leung. "Online scheduling of real-time tasks," In *Proceedings. Real-Time Systems Symposium*, 1988, pp. 244–250.

[21]W. Karamti and A. Mahfoudhi. "Scheduling analysis based on model checking for multiprocessor real-time systems," *J. Supercomput.*, vol. 68, no. 3, 2014, pp.1604–1629.

[22]S. Kato and N. Yamasaki. "Global edf-based scheduling with laxity-driven priority promotion," *J. Syst. Archit.*, vol. 57, no. 5, 2011, pp. 498–517.

[23]K. Kravari and N. Bassiliades. "A survey of agent platforms," *Journal of Artificial Societies and Social Simulation*, vol. 18, 2015.

[24]C. L. Liu and James W. Layland. "Scheduling algorithms for multiprogramming in a hard real-time environment." *J. ACM*, vol. 20, no. 1, 1973, pp. 46–61.

[25]A. Mahfoudhi, W. Karamti, and A. Zaguia., "Scheduling analysis and correction for multiprocessor real-time systems based on multi-agent systems," *International Journal of Applied Engineering Research*, vol. 13, no. 5, 2018, pp. 2368–2374.

[26]V. Manikonda and G. Satapathy. "Cybele: An agent infrastructure for modelling, simulation, and decision support," AgentLink News, 2004, pp. 25–26.

[27]M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos., "A Declarative Model Assembly Infrastructure for Verification and Validation," Tokyo, Japan: Springer, pp. 129–140, 2007.

[28]L. Padgham and M. Winikoff., "Prometheus: A practical agent-oriented methodology," Agent-Oriented Methodologies, 2005, pp. 107–135.

[29]J. Pavón, J. Gómez-Sanz, and R. Fuentes-Fernández., "The INGENIAS methodology and tools," 2005, pp. 236–276.

[30]L. Sha, T. F. Abdelzaher, K-E. Årzén, A. Cervin, T. P. Baker, A. Burns, G. C. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok. "Real-time scheduling theory: A historical perspective." *Real-Time Syst.*, vol. 28, no. 2-3, 2004, pp. 101–155.

[31]S. Han, M. Park, X. Piao, and M. Park. "A dual speed scheme for dynamic voltage scaling on real-time multiprocessor systems," *J. Supercomput*, vol. 71, no. 2, Feb. 2015, pp. 574–590.

[32]M. Šišlák, D.and Rehák, M. Pechoucek, and D. Pavlíc ˇek, "Deployment of a-globe multi-agent platform," In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '06, pp. 1447 – 1448, New York, USA, 2006. Association for Computing Machinery.

[33]M. Wooldridge. *An Introduction to MultiAgent Systems*, 2nd edition. Hoboken, USA: Wiley Publishing.

[34]M. Wooldridge, N. Jennings, and D. Kinny., "The gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, 2009, pp. 285–312.