Multi-agent Dynamic Interaction in Simulation of Complex Adaptive Systems

Hantao Hua†, Feng Zhu‡*, Yiping Yao‡ and Wenjie Tang‡
† College of Computer, ‡ College of Systems Engineering
National University of Defense Technology
Changsha, China
e-mail: {ht_hua | zhufeng | ypyao | tangwenjie}@nudt.edu.cn

***Abstract*—Agent-based modeling and simulation is an effective approach to study complex adaptive systems. With the increasing scale of the simulated system, the interactions between autonomous agents become more and more complex. It is difficult to describe the dynamic interaction between agents by model code intuitively. In addition, the static interaction structure in a multi-agent model leads to long running time and more memory resource consumption. Therefore, this paper proposes a method to graphically describe the dynamic interactions between different agents, named Multi-Agent interaction Graph (MAG). MAG takes agent model class as the basic element of graphical composition. The data communication between agent models is established by using subscribe/publish mechanism, and the interaction between agent model instances is accurately determined based on the dynamic attribute filtering algorithm, which is generated by the large models include Large Language Model (LLM) and Large Vision Model (LVM) automatically from the MAG. The transmission of irrelevant communication data between agent model instances is reduced, and the simulation execution time and memory consumption are reduced. Taking two scenarios as case studies, this paper proves that MAG can model the dynamic interaction between agents, and the execution time of different numbers of agents situation is reduced by 20% - 60%, which can effectively support the scalability of the number of agent model instances. Additional scenario experiments were also conducted to demonstrate the stability and generality of the dynamic attribute filtering algorithm for large model generation.**

Keywords-complex adaptive systems; graphical modeling and simulation; multi-agent interaction; automatic generation of dynamic data filter.

I. INTRODUCTION

Agent based modeling and simulation [1] is an effective approach to study Complex Adaptive Systems (CAS), which is a system composed of autonomous, interacting agents and the interactions between agents will change frequently [2]. With the increasing scale of agents in a CAS, the interactions between agents become more and more complex, showing the characteristics of complex interacting structure and diverse interacting behavior [3]. How to describe the complex dynamic inter-actions between these agents intuitively and how to reduce the interaction overhead in CAS simulation execution bring challenges to traditional multi-agent modeling and simulation approach and environment.

On one hand, it is difficult to describe the dynamic interacions between agents by model code intuitively, because of their complicated interacting structure. The graphical composite modeling method is introduced into multi-agent modeling and simulation, which can simplify the CAS modeling and greatly shorten the development time. Graphical composite modeling method has a higher level of abstraction than code and is closer to the problem domain [4]. Beginners or experienced users can intuitively and efficiently compose simpler models through mouse dragging in a visual user interface to create more complex models and shield the details of code programming such as parallel computing, simulation synchronization, and event scheduling in this process, so as to effectively reduce the difficulty of modeling CAS.

On the other hand, with the increasing of the complexity of the simulated system, the number of the agents will grow, and the communication between the agent models will also increase [5]. The execution time of the complex composite model is usually very long, which limits the scalability of the multi-agent system scale [6]. Even in some special cases, the multi-agent CAS model cannot be executed at all, due to the growth of communication. At present, the complex interactive communication between multi-agent models is mostly limited to a few specific simulation scenarios or reduces the accuracy of the model of CAS. In addition, the existing multi-agent interactive filtering methods are often carried out for specific problems, and there is a lack of a general filtering method, takes time and effort to generate an algorithm for each filtering method.

This paper proposes a multi-agent graphical composite modeling method, named MAG, which takes the agent model as the basic graphical element of the CAS multi-agent model, and establishes the data communication between the agent models by means of public/subscribe. In the actual running process of simulation, the dynamic attribute filtering algorithm is used to filter the non-interactive data between agent models, to build a dynamic agent interaction network. It reduces the transmission of irrelevant communication between agent models, and thus reduce the simulation execution time and memory consumption. The main contribution of this paper is to support multi-agent dynamic interaction in CAS from two aspects. One is graphical modeling and the other is communication data filtering.

(1) The MAG, a graphical composite modeling method, is proposed to solve the complexity of modeling multi-agent dynamic interaction, which is more intuitive than code written by general programming language.

(2) With the MAG modeling method, a dynamic attribute filtering algorithm is introduced to reduce the problem of long simulation time and resource consumption caused by irrelevant communication data transmission between agent model instances. Considering the generality of dynamic attribute filtering algorithm, we use Large Language Models (LLM) to implement

the transformation of filtering methods in MAG to C++ code.

The structure of this paper is as follows. Section II introduces the related work. Section III introduces multi-agent interacion graph. Section IV introduces the general data filtering mechanism generated by the large model. In the Section V, we take the multi-aircraft collision avoidance scenario and swarm robots cooperation scenario as examples, proved that MAG modeling method can effectively support the dynamic interaction modeling between agents of CAS, also proved the stability and scalability of our method. Finally, our conclusion will be made with an indication of the future work.

II. Background and Related work

In terms of graphical modeling approach, Petri net [7] constructs a network structure through Place, Transition, Token and other basic elements to analyze the structure of the system. At present, Petri net is constantly developing forward. For example, Aalst et al. [8] proposed a strategy of modeling complex processes based on hierarchical colored Petri net. Staines et al. [9] proposed the method of transforming UML sequence diagram into Petri net, to make use of its rigorous model verification method. Drakaki et al. [10] proposed a dynamic resource allocation method by combin-ing the colored Petri net and agent-based control system. Tang et al. [11] proposed a hierarchical Petri net modeling paradigm to build an aerial collision avoidance model under multi-aircraft scenario. Jamal et al. [12] extended the agent-based mobile Petri net to simulate the mobility, concurrency, and distribution of agents. Hsieh [13] combined multi-agent architecture and Petri net to solve the problem of distributed dynamic scheduling of hospital patients. However, the current Petri net does not support to describe the dynamic interactions between autonomous multi-agent.

Event Graph is a graphical modeling paradigm of discrete events. In order to improve the composability of event graph, Buss et al. [14] introduced the object-oriented modeling ideas on the basis of the event graph modeling paradigm, and proposed the concept of the Listener Event Graph Ob-jects (LEGO). PEG extended the basic event graph to support graphical composite modeling of discrete event simulation models [15]. Barclay et al. [16] proposed the dynamic chain event graph, which can be combined with Semi-Markov chains. However, the extension and optimization of the above event graph modeling paradigms mostly adopt static interaction structure, which leads to more running time and more resource consumption in a multi-agent model.

Zeigler proposed the Discrete Event System Specification (DEVS) [17]. The DEVS specification contains two types of paradigms, the atomic model paradigm for describing the actual models with specific logical functions, and the coupled model paradigm for building existing atomic models into more complex models. In addition, Hamri et al. [18] introduced minimum/maximum delay into the transition function to extend DEVS for the hardware simulation modeling. Camus et al. [19] combined DEVS specification and multi-agent modeling method to solve the problem of multi-model modeling and simulation of complex systems. Kulakowski et al. [20]adopted the DEVS theory to build a multi-agent system, which was used to simulate the crowd evacuation process when a fire occurred. Jarrah et al. [21]studied the interaction behavior modeling among multiple agents based on the DEVS framework. However, the above extension of DEVS theory does not consider the requirements of reducing multi-agent communication consumption.

Similarly, Markovian agents model is a modeling technique for analyzing large-scale distributed interactive systems. In this model, each agent is treated as a continuous time Markov chain (CTMC) whose behavior is influenced by interactions with other agents [22]. Therefore, the focus of the model is naturally placed on the state itself and the transition probability between states, rather than the dynamic interaction process behind the state transition. Multiformalism modeling approach combines multiple modeling languages and tools to deal with different aspects of complex systems [23]. It pays more attention to the coordination of paradigm, so it is suitable for those complex systems that are difficult to be fully described by a single modeling language, rather than focusing on the dynamic change process of the interaction process between states.

Hybrid hierarchical modeling theory combines models with different description specifications to realize hierarchical hybrid modeling of complex systems. The Ptolemy system developed by Berkeley University [24]was designed under this theory. The Ptolemy model was described by using a Finite State Machine (FSM) with ports. Each state in Ptolemy can also be described by a FSM with ports, to support the construction of application system in a hierarchical way [25]. Flexible Analysis, Modeling and Exercise System (FLAMES) [26] is a commercial hierarchical simulation integration environment, which supports systems and engineering analysis, test and evaluation, training, mission planning and deduction, etc. MathWorks [27]proposed a hybrid dynamic modeling method based on the combination of Simulink and state flow. However, Ptolemy, FLAMES and Simulink were difficult to solve the problem of long running time and more resource consumption caused by irrelevant communication data transmission between multi-agent models.

In the aspect of dynamic interaction modeling, dynamic structure [28] is listed as one of the key challenges of co-simulation, which can be used to study self-organizing cluster system. Uhrmacher [29] constructed a formal theory based on that DEVS is independent to the specific implementation to support the model to adapt to its own interaction structure and behavior and implemented this theory on the James platform. Barros [30] proposed a formal theory, HFSS, which aims to represent hierarchical and modular hybrid systems with dynamic structures. HFSS is a framework for merging components built in different patterns so that a more efficient and easier to understand model can be constructed by changing the structure of the network. Dormido [31] considered the modeling problem of variable-structure hybrid power system. By proposing a new algorithm and using the existing object-oriented hybrid system modeling language, the model was transformed into a model suitable for description and simulation. Hu [32] specifically

discussed the capabilities of the variable structure, structure change, and interface change in the modeling and simulation environment based on DEVS. They discussed the operation of structure change and interface change, and defined the operating boundaries respectively. Most of the above variable-structure modeling methods are based on DEVS or DEVS-like theory, which can describe the dynamic interaction process of multi-agents, but they lack consideration for the optimization of data communication among multi-agents. Fishwick [33] defines hyper-modeling as a general theory for linking system models and their components, but hyper-model is more concerned with interactions within the model, between different models, and between people and models. As the complexity of the model is increased with the increasing complexity of the actual system, which often results in a long simulation time. Mehlhase [34] proposed a framework for modeling and simulating the variable structure models by using hybrid decomposition method in general simulation environment. This method integrates three tools including Dymola, OpenModelica and Matlab/Simulink, and solved the problem of long simulation time by changing the set of equations during simulation running, which reduces the accuracy of CAS models. In addition, the existing multi-agent interactive filtering methods are often carried out for specific problems, and there is a lack of a general filtering method, takes time and effort to generate an algorithm for each filtering method.

Large language models has advanced remarkably in recent years with the advent of pretrained Transformers [35] such as BERT [36] and GPT [37]. With its scaled to hundreds of billions of parameters and started to display early signs of artificial general intelligence, their applications have also transcended text processing, also attempts to bridge the gap between natural langauge processing (NLP) domain and simulation domain on the topic of LLMs applications. LLMs and in-context learning have been applied to code-generation tasks. Chain-of-Thought (CoT) prompting [38], where a language model is prompted with in-context examples of inputs, chain-of-thought rationales (a series of intermediate reasoning steps), and outputs, has shown impressive abilities for solving structured output problems like code-generation tasks. While CoT relies on the ability of LLMs to both generate a reasoning path and execute it, it is not yet possible to translate the agent interaction diagram directly into the executable code. However, although the traditional method can directly convert the agent interaction diagram into code, it needs to be re-modeled for each task and cannot be generalized to general tasks.

The limitation of LLM is that it can only deal with plain text type data and cannot deal with more complex multimodal data types. The task requirements in the field of computer vision are similar to those in the field of natural language processing, and the algorithm of instruction adjustment can be introduced into the field of computer vision or multimodal tasks. LLaVA [39] is an end-to-end trained large-scale multimodal model that connects a visual encoder and a language model for multimodal tasks such as general vision and language understanding.

In a word, in addition to the above current related work, a new method is needed. On one hand, it solves the problem of modeling complexity of dynamic interaction between agents intuitively. On the other hand, it solves the problem of long running time and more memory consumption caused by irrelevant communication data transmission of dayamic interacion. In addition, the generality and efficiency of the new method should be guaranteed.

III. Multi-agent Interaction Graph

The MAG is designed to support the modeling of the dynamic interaction between agents. It uses the agent model class instead of agent model instance as the basic graphical modeling element. An attribute filter should be configured between each pair of agent model class. After the graphical multi-agent model is built, it will be automatically translated into the executable model of *LP* (Logical Process) paradigm. During the simulation execution time, the agent model instances are generated dynamically according to the simulation scenario, and the interaction between the agent instances is determined dynamically based on the calculation results of the attribute filter.

A. Graphical Representation of Agent Model

To describe the agent model graphically, a port diagram is used to represent the agent model class, as shown in Figure 1. The ports in the diagram are mainly responsible for establishing the communication channel between agent models. The port *p1* represents the input port of the agent model, and the port *p2* represents the output port of the agent model. The behavior logic of the agent model can be graphically represented as an extended Petri Net [40]. The $data$ components are used to store the state data of the agent model. The state data can be used as the input data to a $function$ component. The $function$ components are employed to indicate the computing functions, which changes the state data of the agent model.. The $link$ component is used to represent the control flow and the data flow. A directed arc from $data$ component to $function$ component means the data component is an input of the $function$ component. A directed arc from $function$ component to $data$ component means the data component is an output of the $function$ component.

To facilitate users to understand the structure provided by the agent model, it is necessary to provide the metadata information of the agent model class, as shown in the right side of Figure 1. *ClassName* represents the class name of the agent model, and each agent model will be automatically transformed into a C++ class. *AttributeList* denotes the set of state variables of the agent model. *EventList* represents the set of events in the agent model. The processing of each event will cause the agent state to change instantaneously. *PortList* denotes the port set of the agent model for external interaction. The agent model can communicate with other agent models through input/output port. *MapList* represents the set of the pair $< event, port >$. The processing of an event in an agent will not only change its own state, but also trigger an interaction between the agent model and another agent models. The interaction here is

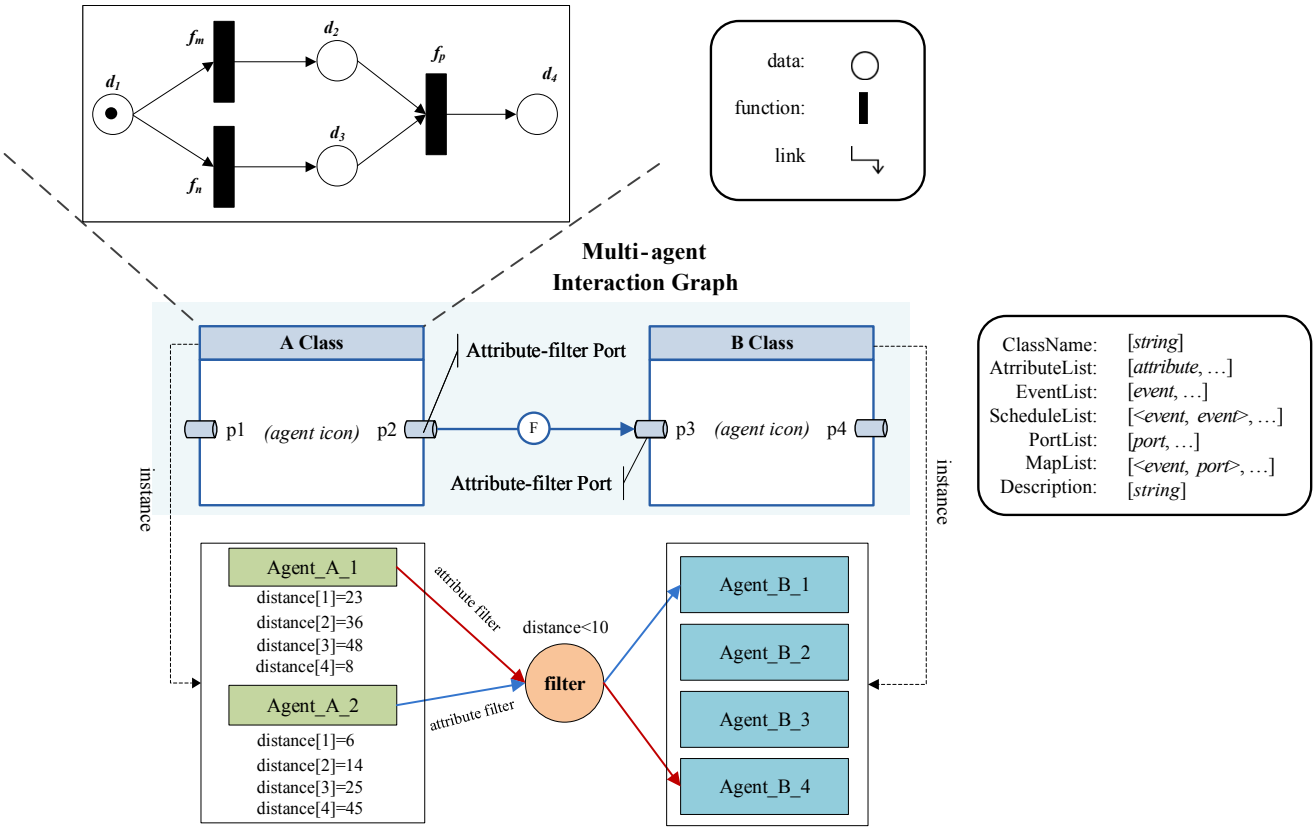


Figure 1. Graphical representation of an agent model and its interacion.

realized through port communication. Description represents the behavior rules.

B. Structure of Multi-agent Interaction Graph

MAG describes the dynamic interaction between agent model instances shown in Figure 1. The connection with attribute filter F between the ports of two agent models *Agent_A* and *Agent_B* indicates that there may be interactions between them. The attribute filter can also not be configured, which means that there is always interaction between the two agent models. Before the simulation execution, the attribute filtered expression set should be configured on the attribute filter F. Several filtered expressions can be established for the same attribute of the agent model. The filtered expression establishes the criterion of pass/reject on the attribute. To pass a filter, at least one filtered expression in that filter should be satisfied. Only by passing a certain attribute filtered expression configured by the agent model, it is considered that the interaction conditions between the agent model instance and the corresponding agent model instance can be satisfied, so as to realize the communication between the agent instances. In the process of simulation execution, several instances of the agent model class will be generated, and the interaction between specific agent model instances is determined by the attribute filter F.

As shown in Figure 1, the agent model class *Agent_A* connects with the agent model class *Agent_B*, which indicates that the instance of agent model class *Agent_A* perhaps interact with the instances of agent model *Agent_B*. It is supposed that there are two instances of class *Agent_A* and four instances of class *Agent_B* will be generated according to the simulation scenario. The communication from the agent model instances of the class *Agent_A* to the agent model instances of the class *Agent_B* is determined by the attribute filtered expression $distance < 10$ which is configured on the filter. Supposing that the positions of all the agent model instances of class *Agent_B* remains unchanged, when the position of the agent model instances of class *Agent_A* is updated, the variable distance will be calculated. Therefore, by checking whether the expression $distance < 10$ is true, which two agent moddel instance can communicate with each other is dtermined. For example, the model instance *Agent_A_1* is able to interact the model instance *Agent_B_4*. Similarly, the model instance *Agent_A_2* is able to interact to the model instance *Agent_B_1*.

IV. Automatic Filter Generation

An agent model can be described as an extended Petri net [40], each place is used to store the state variables of the agent model, and each transition is always labeled by an event. The dynamic interaction between the agents can be modelled by MAG. In order to enable the execution of a CAS model on existing Discrete Event Simulation (DES) platforms, we map

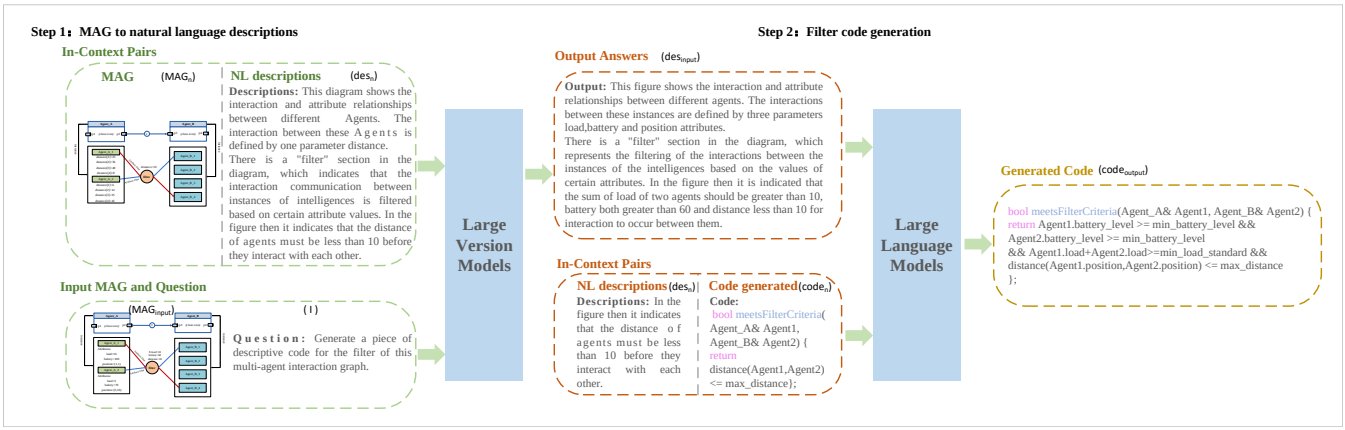


Figure 2. Dynamic data filtering automated generation process.

an agent to a *LP* and mapping the interactions between agents to update attributes between *LPs* [41].

Previous work has implemented the transformation of Agent elements from MAG diagrams to code, in the process of translating MAG to DES code, we search all the graphical elements and convert the graphical element into discrete event simulation code. For each agent model element, we build a *LP* class in C++ object-oriented language. An agent model is translated into a *LP* class which is composed of an initial method, several event procedures, and a series of variables. Each initial method sets the initial value for the variables of the *LP* and then schedule some necessary events for simulation execution. A *LP* changes the values of the variables that describe the agent state through processing events. For each filter element, it is needed to search for all agents associated with a filter and make a connection with them. After that, in the simulation execution, the code for scheduling interactive events will be executed according to the calculation of the filtered expression in the filter.

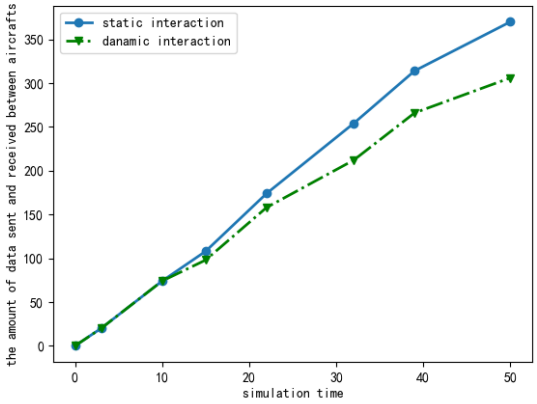
It is not easy to implement the attribute filtering method in MAG and the parameter in agent into code expressions, especially considering that there may be many different expressions of MAG in practical applications. It is difficult to find a general filter code expression generation method. The generation of filters is often closely related to the content presented in MAG. LVM can describe the information presented in MAG as natural language, but LVM does not perform well on natural language code generation tasks. Similarly, some common LLMS work well for the task of generating code in natural language, but do not take pictures as their input. For the text of these given output styles of code, because of the hallucination of LLM, the direct use of LLM to generate will bring a certain chance of misformatted output. In-Context Learning (ICL) is brought to reduce the probability of hallucination, that is, several input-output generation examples are placed on the input side, and then the problem description and input are used as prompt for LLM/LVM. This has proven to be an efficient means of structural text generation based on LLM/LVM [42].

We use large models (including LLMs and LVMs) and CoT Prompting to convert MAG into filter code expressions. Given the picture form of MAG, we break the problem down into a step-by-step form based on CoT prompt, as follows: LVM based on ICL is used to convert MAG into text description close to natural language, including input parameters and discriminant logic of filter expression, and LLM is used to convert generated natural language into corresponding filter code. The context and CoT prompt process is shown in Figure 2. In Figure 2, for the MAG shown in Figure 1, first complete the generation of in-context examples of MAG-description on the manually configured MAG sample set as prompt input to the LVM. The description for filter generating the code of MAG is obtained, includes the input parameters and the discriminant logic of the code. The same ICL method is used in the LLM generation code phase, merging LVM generated descriptions with in-context examples as prompt input to the LLM. These examples are manually written and can typically be constructed without an accompanying graph. Then, it will generate a program (right of Figure 2) that can be executed which is based on the input MAG(s) to perform the described filter task.

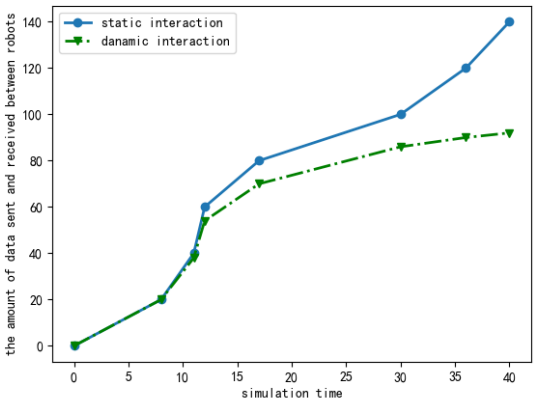
$$LVM(I, pairs < MAG_1, des_1 >, pairs < MAG_2, des_2 >, \cdots, pairs < MAG_n, des_n >, MAG_{input}) \rightarrow des_{input} \quad (1)$$

$$LLM(I, pairs < des_1, code_1 >, pairs < des_2, code_2 >, \cdots, pairs < des_n, code_n >, des_{input}) \rightarrow code_{output} \quad (2)$$

We showed in (1) and (2) for the process of generating code from MAG using LLM in combination with LVM. Here, $pairs\langle x, y\rangle$ is the ICL to identifie the example MAG-description pair or description-code pair, as shown in the in-context pairs above step 1 in Figure 2. Where I represents the definition of the task, MAG_{input} represents the input we want to get the result, as shown input MAG and question below step 1 in Figure 2. des_{input} represents the intermediate module

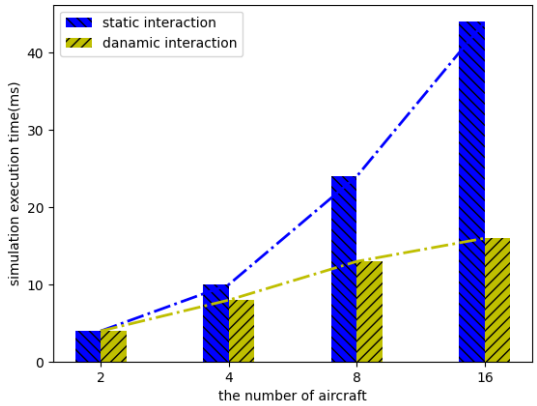


(a) Aircraft Collision Avoidance Scenario

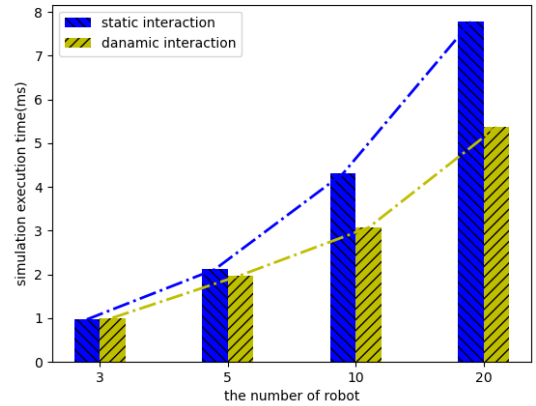


(b) Swarm robots Cooperation Scenario

Figure 3. The total amount of communication data.

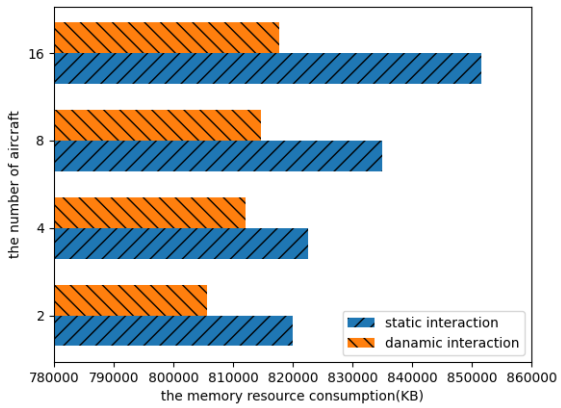


(a) Aircraft Collision Avoidance Scenario

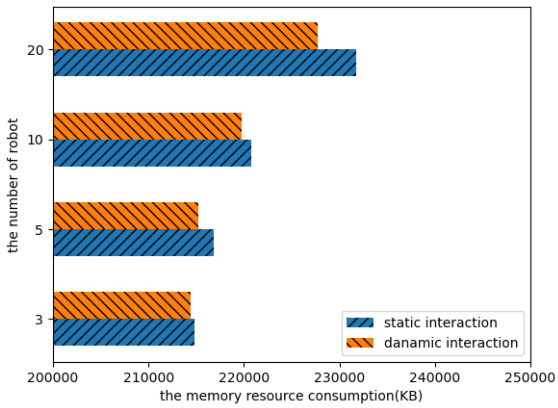


(b) Swarm robots Cooperation Scenario

Figure 4. Execution time of MAG-based model.



(a) Aircraft Collision Avoidance Scenario



(b) Swarm robots Cooperation Scenario

Figure 5. Memory consumption of the MAG-based model.

for filter code generation, the same way to generate the filter code we end up with like step 1, we can get the result code $code_{output}$.

V. Experiments and Analysis

Our work provides a general, efficient and scalable multi-agent simulation framework. We test some typical application scenarios of multi-agent simulation, including aircraft collision avoidance scenario with less inter-agent communication interaction and robot collaboration scenario with more frequent inter-agent communication interaction. We now describe these scenarios, their evaluation settings, and their results for analyzing.

A. Test Scenarios Introduction

Aircraft Collision Avoidance Scenarios. In the airspace near the civil airport, with the increasing of airspace density, collision avoidance between aircraft is considered to be a key problem affecting flight safety. Because the time of aircraft collision is very short, it is necessary to predict the flight path of the aircraft as soon as possible to detect the possible collision risk, so as to guarantee the multi-aircraft flight. Agent-based modeling and simulation is very suitable for analyzing multi-aircraft collision avoidance scenarios. In this paper, we assume that (1) before any two aircraft enter the adjacent airspace, there is no need to exchange position data between aircraft; (2) the height of the collision cylinder which are twice as long as the vertical size of the aircraft; and (3) two aircraft may collide when they are encountering each other at a certain angle, and they will not collide again after the collision is released. Under this assumption, the modeling for multi-aircraft collision avoidance in this scenario involves the following interactions: (1)when any two aircraft enter the adjacent airspace, there may be collision between them. In order to avoid collision, these aircraft often need to exchange position data and adjust the corresponding flight direction and angle in time; and (2) when any two aircraft are free of collision risk, the position data sharing between aircraft can be cancelled to reduce the communication between aircraft models.

This paper takes Tang-Piera multi-aircraft collision problem [43] as a segggd on GMAS, to verify the running time, memory cost and scalability of the model based on the MAG modeling method and the dynamic attribute filtering algorithm based on LLMs.

We use three agent model classes to build a four-aircraft collision avoidance model. Since there is no risk of collision avoidance between *aircraft0* and *aircraft3*, they can be instantiated by the same agent model class *AircraftA*. The only difference between them is their initial position and speed. *aircraft1* is generated by the agent model class *AircraftB* and *aircraft2* is generated by the agent model class *AircraftC*. Since the collision avoidance logic in each aircraft is the same, the same filter can be used for communication between aircraft. Its filter expression can be set to horizontal $distance < 88$ m, which also means that when the horizontal distance between two aircraft is within this range, the position information of the other aircraft will be sent or received. Therefore, a multi-aircraft dynamic interaction model is constructed graphically through GMAS.

Swarm robots cooperation Scenarios. In search and rescue environments, swarm robots are often used to locate people or transport materials. These robots often have different functions, such as sonar detection, infrared thermal imaging and robotic arm operation, they collaborate with each other to complete more complex tasks. Since the scene is often more complex and difficult to predict, multi-agent modeling and simulation of robot behavior has become a feasible and efficient way. In this paper, we assume that (1) there are three kinds of robots in the scenario, which have the functions of sonar detection, infrared thermal imaging and robotic arm operation respectively; (2) all sonar detection and infrared thermal imaging robots will search after receiving the search task, and if they find trapped people, they will cooperate with the robotic arm operator to complete the rescue task; and (3)each probe robot works with a maximum of one arm-operated robot. Under this assumption, the modeling for swarm cooperative robots in this scenario involves the following interactions: (1)when the probe robot finds the trapped person, it will look for and cooperate with the nearest robotic arm robot that is free and meets the demand of power and load, which requires the information interaction and communication between the probe robot and all robotic arm robots; and (2)when the probe robot finds the robot arm that works with it, the data sharing between the robots is no longer required for reducing the communication between robot models. Similarly, we use GMAS [40] environment and implement the dynamic attribute filtering algorithm at the swarm robots cooperation scenarios to verify the running time, memory cost and scalability of the model based on the MAG modeling method and the dynamic attribute filtering algorithm based on LLMs.

We use three agent model classes to build a swarm robots cooperation rescue model, since there is no needs for communication between each kinds of robots, each kinds of robots can be instantiated by one agent model class like *RobotA*. The only difference between them is their pamameters like location, remain power, etc. The probability of a "life-form found" event is poisson distributed as the robot keeps searching when no result is found. The probability of "finding a life form" event is Poisson distributed. After finding a life form, the probe robot will search for the nearest eligible robot arm to cooperate in the rescue mission. Since the same interaction parameters are required for each type of probe robot and robotic arm robot in the process of seeking cooperation, the same communication filter can be applied. Information, such as the current status, remaining power and load capacity of the robot is filtered. When the above conditions of the two robots meet the standard, the location data exchange between the two robots is started to achieve collaborative rescue. Figure 2 illustrates the large model-based MAG-to-filter code generation process for the swarm robot collaboration scenario. The generated code will be directly used in the discriminative expression of the filter.

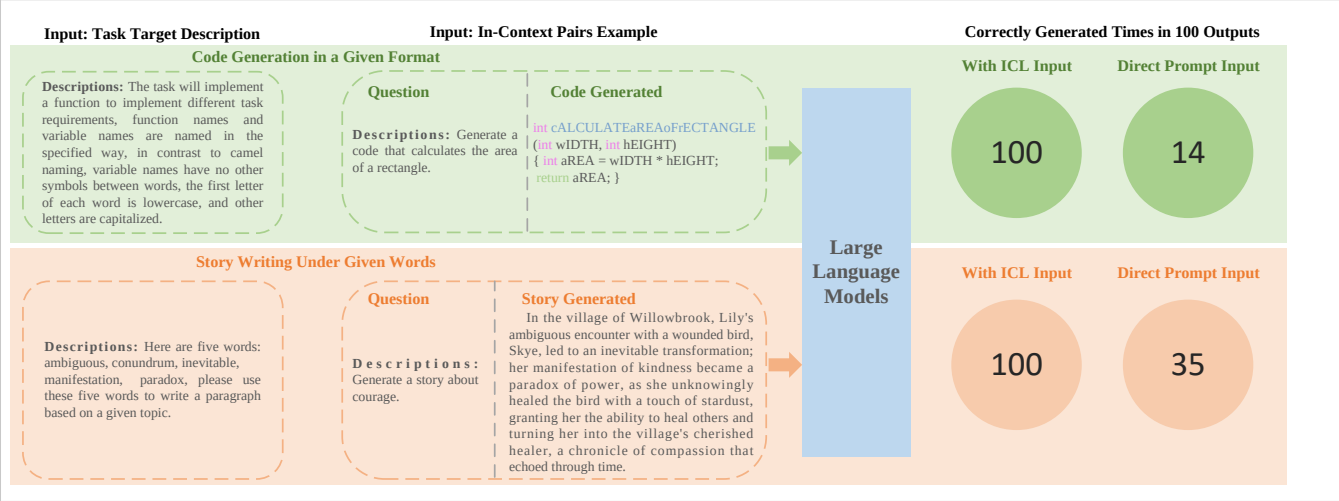


Figure 6. Dynamic data filtering automated generation process.

B. Scenarios Realization Analysis

Aircraft Collision Avoidance Scenarios. We simulated the collision scenario of Tang-Piera four-aircraft in a very short period of time. Four aircraft were both into the adjacent airspace, and successively entered the scene of *collision 1*, *collision 2* and *collision 3*. When the simulation time $T \in [0, 2]$, each two of the four aircraft are in the adjacent airspace, they need to share position data with each other to determine whether they are in collision with an aircraft. When the simulation time $T \in [3, 10]$, *aircraft 0* and *aircraft 1* will encounter the *collision 1*, they will continue to share position data.When the simulation $T > 10$, the *collision 1* is released, and the interaction between them can be cancelled dynamically. When the simulation time $T \in [15, 22]$, *aircraft 0* and *aircraft 2* will encounter the *collision 2*, and they need to continue to maintain interaction.When the simulation $T > 22$, the *collision 2* is released, and the interaction between them can be dynamically disconnected. When the simulation time $T \in [32, 39]$, *aircraft 2* and *aircraft 3* will encounter the *collision 3*, and they need to continue to maintain interaction. When the simulation time $T \in [40, 50]$, the *collision 3* threat is removed, and the four aircraft return to the predetermined trajectory to continue to fly, and the interaction between *aircraft 2* and *aircraft 3* can be cancelled dynamically. We can see that the altitude difference between the two aircraft will collide within [0,10].

Figure 3(a) compares the amount of communication data sent and received between aircraft in two different situations of static interaction (SI, data interaction is always maintained during simulation execution) and dynamic interaction (DI, when the collision is released, the interaction between them can be disconnected dynamically). We can find that when the simulation execution is finished, the total amount of communication between the four aircraft is reduced by 17%. **Swarm robots cooperation Scenarios.** We also simulated the swarm robots cooperation scenario in a period of time. Ten robots of each type are initialized to move randomly on a given map, their power gradually decreases with its working time. When the simulation time starts, each robot performs its own detection task separately and does not need to communicate with each other. While a probe robot detects a life form, information such as position and load is shared with idle robotic arm robots in the area with power above a threshold instead of sharing it as a broadcast to reduce data communication. The detecting robot selects the nearest robot that meets the load requirements to cooperate and no longer shares data information with other robots. The above two scenarios are repeated until the task is completed. Figure 3(b) shows the total amount of data communicated under the static and dynamic interaction algorithms. It can be clearly seen that the amount of communication data generated by the dynamic interaction using the filtering-based algorithm produces a significant decrease, and more so as the simulation time progresses, reaching 34% of the optimized amount of communication data at the end of the simulation.

C. Scalability, Stability and generality of our system

Figure 4 compares the execution time of MAG-based model under two different situations, one is static interaction and the other is dynamic interaction. With the increasing of the number of agent, the simulation running time increases when the interaction is not disconnected. When the interaction is disconnected dynamically, the execution time of multi-agent model in 2-16 aircraft situation is reduced by 20%-60% compared with the former, and the execution time of multi-agent model in swarm robot cooperation scenario reduced by approximately 30%. Figure 5 compares the memory resource consumption of the MAG-based model under two different situations: static interaction and dynamic interaction. With the increasing of the number of agent, the consumption of model memory resources of multi-agent model in 2-16 aircraft situation is reduced by 1.8%-4%, and is reduced by 0.8%-5.5% in swarm robot cooperation scenario. That's because the sharing data is only position data(x,y) in this multi-aircraft

collistion avoidance model and more attribute data needs to share in the swarm robot model. In summary, it is can be concluded that MAG-based model has good scalability.

The results obtained from large model-based methods often bring problems of interpretability and consistency, and the use of ICL will enhance the stability and consistency of the results. For several simple fixed-style text generation tasks, we use description-text pairs of corresponding styles as ICL pairs, and test whether the results meet the requirements, while using a non-ICL approach (direct input of prompt) as a comparison. Figure 6 shows a description of the different text generation tasks, the ICL pairs used, and the number of times the target text was correctly generated over 100 tests with different inputs. The test environment is Qwen2-7B. The results show that ICL can greatly improve the stability of the output results under the text generation task, so it can be better used in the code generation task under the given format required in this paper.

We also evaluated the accuracy of the filter code generation, constructing different forms of MAG with filter execution logic for validation. To verify the stability and generality of our algorithm, we tested the filter code generation results using different combinations of large models (LLaVa [39]+LLaMa3 [44] and Qwen-VL [45]+Qwen2 [46]), while ablation experiments were performed to verify the necessity of implementing the model architecture in this way. We tested the filter code generation effect with LLaVa+LLaMa3, Qwen-VL+Qwen2 and LVM-only on 100 randomly generated MAGs (manually filtered to ensure the diversity of scenarios and filtering mechanisms), and the test results are shown in Table I.

TABLE I. **NUMBER OF SUCCESSFUL FILTER CODE GENERATION FOR DIFFERENT MODELS COMBINATION**

Large models (portfolios) used	Successful generation number
LLaVa+LLaMa3	78
Qwen-VL+Qwen2	74
LLaVa-only	17

Here, "successful generation" means that the generated code can be directly used in the filter and realizes the correct filtering function, which shows the stability and generality of the combined filter code generation algorithm using LLM and LVM.

VI. CONCLUSION AND FUTURE WORK

Multi-agent modeling and simulation is an effective means to study complex adaptive system. The traditional static interaction structure leads to the long running time and resource consumption, then this paper proposes a MAG modeling method to graphically describe the dynamic interaction between agent models. In MAG, the communication between agent models is established by using publish/subscribe, and the interaction between agent instances is accurately determined based on the dynamic attribute filtering algorithm. The aircraft collision avoidance experiments in multi-aircraft scenario and the swarm robots cooperation scenario show that MAG can model the dynamic interaction between agents of CAS, reduce the transmission of irrelevant communication data between multi-agent instances, and reduce the simulation execution time and resource consumption. In addition, it shows good scalability. Different combinations of large models and ablation tests have proven the stability and versatility of the system.

In future work, we will build more complex simulation models based on GMAS to verify the effectiveness and efficiency of the MAG-based modeling and simulation method.

REFERENCES

[1] L. F. M. Bristow and K. W. Hipel, "Agent-based modeling of competitive and cooperative behavior under conflict", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7, pp. 834–850, 2014.

[2] C. M. Macal and M. J. North, "Tutorial on agent-based modelling and simulation", *Journal of Simulation*, vol. 4, no. 3, pp. 151–162, 2010.

[3] R. Dekkers, "Complex adaptive systems", *Applied Systems Theory*, pp. 211–233, 2017.

[4] T. T. M. Haeusler and J. Kessler, "Chronosphere: A graph based emf model repository for it landscape models", *Software and System Modeling*, vol. 18, no. 4, pp. 1–40, 2019.

[5] R. M. Fujimoto, "Research challenges in parallel and distributed simulation", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 26, no. 1, pp. 1–29, 2016.

[6] A. Rousset, B. Herrmann, C. Lang, and L. Philippe, "A survey on parallel and distributed multi-agent systems for high performance computing simulations", *Computer Science Review*, vol. 22, pp. 27–46, 2016.

[7] J. Haas, "Stochastic petri nets for modeling and simulation", in *Proceedings of the 2004 Winter Simulation Conference*, 2004, pp. 101–112.

[8] W. Aalst, C. Stahl, and M. Westergaard, "Strategies for modeling complex processes using colored petri nets", in *Transactions on Petri Nets and Other Models of Concurrency VII* (Lecture Notes in Computer Science), Lecture Notes in Computer Science. Springer, 2013, vol. 7480, pp. 6–55.

[9] S. Staines, "Transforming uml sequence diagrams into petri net", *Journal of Communication and Computer*, vol. 10, no. 1, pp. 72–81, 2013.

[10] M. Drakaki and P. Tzionas, "Modeling and performance evaluation of an agent-based warehouse dynamic resource allocation using colored petri nets", *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 7, pp. 736–753, 2015.

[11] J. Tang and F. Zhu, "Graphical modelling and analysis software for state space-based optimization of discrete event systems", *IEEE Access*, vol. 6, pp. 1–15, 2018.

[12] M. Jamal and N. Zafar, "Extending agent-based mobile petri nets with access control", in *2017 IEEE International Conference on Communication, Computing and Digital Systems (C-CODE)*, IEEE, 2017, pp. 133–138.

[13] F. Hsieh, "A hybrid and scalable multi-agent approach for patient scheduling based on petri net models", *Applied Intelligence*, vol. 47, no. 4, pp. 1–19, 2017.

[14] A. Buss and C. Blais, "Composability and component-based discrete event simulation", in *Proceedings of the 2007 Winter Simulation Conference*, 2007, pp. 694–702.

[15] B. Wang, B. Deng, and F. Xing, "Partitioned event graph: Formalizing lp-based modelling of parallel discrete-event simulation", *Mathematical and Computer Modelling of Dynamical Systems*, vol. 21, no. 2, pp. 153–179, 2014.

[16] L. Barclay, R. Collazo, J. Smith, *et al.*, "The dynamic chain event graph", *Electronic Journal of Statistics*, vol. 9, no. 2, pp. 2130–2169, 2015.

[17] B. P. Zeigler, G. Kim, and H. Praehofer, *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic Press, 2003.

[18] A. Hamri, N. Giambiasi, and C. Frydman, "Min-max-devs modeling and simulation", *Simulation Modelling Practice and Theory*, vol. 14, pp. 909–929, 2017.

[19] B. Camus, C. Bourjot, and V. Chevrier, "Combining devs with multi-agent concepts to design and simulate multi-models of complex systems", in *2015 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (DEVS)*, IEEE, 2015, pp. 85–90.

[20] A. Kulakowski and B. Rogala, "Agent devs simulation of the evacuation process from a commercial building during a fire", in *2017 Conference on Computer Science and Information Technologies (CSIT)*, IEEE, 2017, pp. 270–279.

[21] M. Jarrah, B. Zeigler, C. Xu, *et al.*, "A multi-agent simulation framework to support agent interactions under different domains", in *2015 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES)*, IEEE, 2015, pp. 211–223.

[22] A. Bobbio, D. Cerotti, M. Gribaudo, M. Iacono, and D. Manini, "Markovian agent models: A dynamic population of interdependent markovian agents", in *Seminal Contributions to Modelling and Simulation: 30 Years of the European Council of Modelling and Simulation*, K. Al-Begain and A. Bargiela, Eds. Cham: Springer International Publishing, 2016, pp. 185–203, ISBN: 978-3-319-33786-9. DOI: 10.1007/978-3-319-33786-9_13.

[23] J. Yang, H. Peng, W. Zhou, J. Zhang, and Z. Wu, "A modular approach for dynamic modeling of multisegment continuum robots", *Mechanism and Machine Theory*, vol. 165, p. 104 429, 2021, ISSN: 0094-114X. DOI: https://doi.org/10.1016/j.mechmachtheory.2021.104429.

[24] C. Ptolemaeus, *System Design, Modeling, and Simulation: Using Ptolemy II*. Berkeley: Ptolemy.org, 2014, ISBN 978-1-304-42106-6.

[25] M. Kielar, O. Handel, and H. Biedermann, "Concurrent hierarchical finite state machines for modeling pedestrian behavioral tendencies", *Transportation Research Part B: Methodological*, vol. 70, pp. 576–584, 2014.

[26] T. Corperation, "Flames makes complex systems analysis simple", 2015, [Online]. Available: http://www.ternion.com/print/FLAMES.pdf.

[27] A. Rajhans, S. Avadhanula, A. Chutinan, *et al.*, "Graphical modeling of hybrid dynamics with simulink and stateflow", in *2018 21st International Conference on Hybrid Systems: Computation and Control (HSCC)*, IEEE, Porto, Portugal, 2018, pp. 247–252.

[28] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey", *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, 2018.

[29] A. M. Uhrmacher, "Dynamic structure in modeling and simulation: A reflective approach", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 11, no. 2, pp. 206–232, 2001.

[30] F. J. Barros, "Dynamic structure multiparadigm modeling and simulation", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 13, no. 3, pp. 259–275, 2003.

[31] A. Urquia and S. Dormido, "Object-oriented description of hybrid dynamic systems of variable structure", *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 79, no. 9, pp. 485–493, 2003.

[32] X. Hu, B. P. Zeigler, and S. Mittal, "Variable structure in devs component-based modeling and simulation", *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 80, no. 2, pp. 91–102, 2005.

[33] P. Fishwick, "Hypermodelling: An integrated approach to dynamic system modelling", *Journal of Simulation*, vol. 6, no. 1, pp. 2–8, 2012.

[34] A. Mehlhase, "A python framework to create and simulate models with variable structure in common simulation environments", *Mathematical and Computer Modelling of Dynamical Systems*, vol. 20, no. 6, pp. 566–583, 2014.

[35] A. Vaswani *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805.

[37] T. B. Brown *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165.

[38] J. Wei *et al.*, *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903.

[39] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning", in *Advances in Neural Information Processing Systems*, A. Oh *et al.*, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 34 892–34 916.

[40] F. Zhu and J. Tang, "Graphical composite modeling and simulation for multi-aircraft collision avoidance", *Software and Systems Modeling*, 2020.

[41] F. Zhu, Y. Yao, W. Tang, and J. Tang, "A hierarchical composite framework of parallel discrete event simulation for modelling complex adaptive systems", *Simulation Modelling Practice and Theory*, vol. 77, pp. 141–156, 2017.

[42] Q. Dong *et al.*, *A survey on in-context learning*, 2024. arXiv: 2301.00234 [cs.CL].

[43] J. Tang, M. A. Piera, and T. Guasch, "Colored petri net-based traffic collision avoidance system encounter model for the analysis of potential induced collisions", *Transportation Research Part C: Emerging Technologies*, vol. 67, pp. 357–377, 2016.

[44] H. Touvron *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[45] J. Bai *et al.*, *Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond*, 2023. arXiv: 2308.12966 [cs.CV].

[46] J. Bai *et al.*, *Qwen technical report*, 2023. arXiv: 2309.16609 [cs.CL].