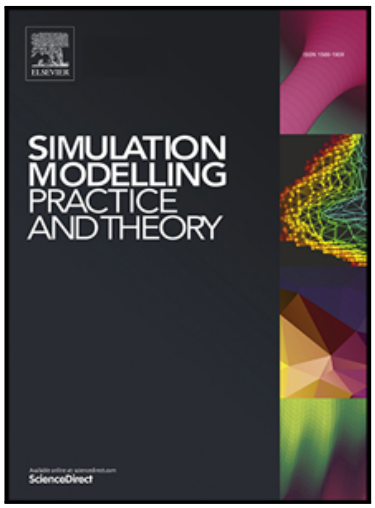

Model Maturity-Based Model Service Composition in Cloud Environments

Ying Liu , Lin Zhang , Yongkui Liu , Yuanjun Laili , Weicun Zhang

PII: S1569-190X(21)00096-4
DOI: https://doi.org/10.1016/j.simpat.2021.102389
Reference: SIMPAT 102389

To appear in: *Simulation Modelling Practice and Theory*

Received date: 21 February 2021
Revised date: 2 August 2021
Accepted date: 3 August 2021

Please cite this article as: Ying Liu , Lin Zhang , Yongkui Liu , Yuanjun Laili , Weicun Zhang , Model Maturity-Based Model Service Composition in Cloud Environments, *Simulation Modelling Practice and Theory* (2021), doi: https://doi.org/10.1016/j.simpat.2021.102389

Model Maturity-Based Model Service Composition in Cloud Environments*

Ying Liu [1, 2, 3], Lin Zhang [1, 2, 3*], Yongkui Liu [4], Yuanjun Laili [1, 2, 3], Weicun Zhang [5]

[1] School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing, China
[2] Engineering Research Center of Complex Product Advanced Manufacturing Systems Ministry of Education, Beijing, China
[3] Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, Beijing, China
[4] School of Mechano-Electronic Engineering, Xidian University, Xi'an, China
[5] School of Automation and Electrical Engineering University of Science and Technology Beijing, Beijing, China

Abstract: With the development of cloud computing (CC), service-oriented architecture (SOA), and container technology, modeling and simulation (M&S) resources, such as simulation software and different sorts of models, can be shared and reused in a cloud environment. Modeling and Simulation as a Service (MSaaS), as a new paradigm, supports sharing simulation models or modeling tools and has enabled a wide range of model reuse. However, reusing or combining some immature models may result in inefficient M&S activities or even false simulation results. To make sure the appropriate reuse and composition of simulation models in cloud environments, which is also termed as model service composition for simulation (MSCS), this paper incorporates model maturity with service cooperation as a metric to evaluate the quality of model composition in cloud. Then, as a multi-objective optimization problem with multiple constraints, the MSCS problem and its process are described in detail. To solve the MSCS problem, a novel evolutionary algorithm named CA-AO-NSGAII is proposed. In the algorithm, adaptive crossover and mutation operators, as well as probabilistic initialization are developed. Furthermore, a half-local search algorithm in an elitist mechanism is designed for efficient decision-making. To validate the performance of CA-AO-NSGAII, experiments with respect to four different cases are conducted. Results show that the proposed method for addressing MSCS issue is effective and feasible.

Key words: Modeling and Simulation (M&S), Cloud Computing, MSaaS, Model Service Composition for Simulation (MSCS), Model Maturity, Evolutionary Algorithm

1. Introduction

In recent years, with the rapid increase of simulation scale and simulation requirements, the number of simulation models and simulation systems is gradually expanding, accompanied by high cost, inefficiency, low quality and reliability in the development of models and simulation systems. Model reuse is an effective way to solve the above problems, and model composition is one of the most used ways to realize model reuse. In order to achieve the composition of models in the simulation field, from the earliest DEVS (Discrete Event System Specification) architecture [1, 2] to the current service-oriented architecture (SOA) for model development [3], the study of model composition has experienced a long time. However, it is still a difficult and hotspot problem. So far, the current trend is that the simulation models are

* Corresponding author email: johnlin9999@163.com.

combined as cloud services to fulfill the different simulation requirements. And there are lots of architectures and patterns to implement the composition and reuse of simulation models with the combination of SOA and cloud computing, such as Web-based Simulation (WBS) [4], Cloud-based Simulation (CSim) [5], Cloud Simulation Platform (CSP) [6], Modeling and Simulation as a Service (MSaaS) [7], Simulation Software-as-a-Service (SimSaaS) [8], etc. These paradigms can unify and effectively manage the decentralized modeling and simulation (M&S) resources, use and share them on-demand at multiple granularities in the form of M&S services, especially model services.

Currently, the most widely used and studied pattern is MSaaS. There are mainly four paradigms in MSaaS: modeling as a service, model as a service, V&V as a service, and simulation as a service. Meanwhile, many platforms with MSaaS patterns are developed to provide coarse-grained and fine-grained M&S services. This paper mainly focuses our research on the paradigm of model as a service, which means that users can share simulation models in the cloud by accessing them through services. As the granularity of services in cloud becomes more and more refined, simulation models become more and more single-function and precise. Meanwhile, selecting reliable, high-quality, and mature model services from the candidate model services and putting them together in a specific process to accomplish a simulation task efficiently, i.e., model service composition for simulation (MSCS), becomes an important issue.

Issues of service composition in cloud manufacturing and other cloud environments exist as well. The difference is that MSCS needs to take into account the state of models at a given moment. Furthermore, the model service composition is usually temporally and spatially consistent, concurrently interactive, and collaborative. In addition, the model service composition needs to maintain a consistent interface between model services and configure the simulation engine to run simulations on one RTI (run time infrastructure) based on time or event. In addition, the credibility of the combined model services also needs to be evaluated. Therefore, there is more need to study the specific processes of MSCS from both functional and non-functional perspectives.

Before composition, model credibility is a metric commonly used to assess the quality of one single model. After composition, it is very difficult to accurately determine the credibility of a combination model based on every single model's credibility. More importantly, model credibility might change as demands change when models are reused in the form of model services. Model maturity [9] is an important complement to evaluate the model quality. It assesses the model's whole lifecycle process (model construction, model use, and model management), and is a suitable metric to evaluate the quality of models in MSCS. It considers the model standardization, portability, scalability, and other features that have an important impact on model use and reuse, which model credibility does not address. Model maturity does not change with demands exactly but changes with time and model evolution, which is well suitable for the QoS evaluation of MSCS. Compared to other common QoS metrics (e.g., time, cost, reliability, etc.), model maturity can highlight the differences between candidate model services, which allows us to identify appropriate combinations of model services more efficiently. Therefore, this paper considers the issue of MSCS based on model maturity from the non-functional perspective.

Due to the diversity and complexity of the simulation process in SoS (system of systems) [10], simulation tasks can be classified into two categories: 1) single simulation service request task (S-SSRT), which can be completed by invoking only one model service. 2) multiple simulation request task (M-SSRT), which has to be completed by invoking several model services to execute simulation together in a particular sequence with one RTI or simulation environment. The optimal model service selection for M-SSRT is more complex than the selection for S-SSRT, because of its characteristics of multi-objectives,

multiple constraints, and multi-candidate sub-services. M-SSRT needs large-scale model service composition and optimal selection (MSCOS) to give an optimal solution. So MSCOS, just as service composition and optimal selection (SCOS) problem [11], is a typical NP-hard problem with dynamic, complexity, and uncertainty characteristics. Consequently, MSCS, as one of the MSCOS issues, is also an NP-hard problem.

However, most research and commercial applications in MSCOS mainly focus on functional composition, such as architectures, middleware, and simulation tools or platforms for model service composition. Furthermore, most of the service composition and optimal-selection approaches neglect the relationship among model services and do not consider the characteristics of the model composition in simulation. There are still three main issues that need to be addressed, which are as follows: 1) The process of simulation-oriented model service composition is different from the process of service composition in cloud manufacturing or other cloud computing services. There are no explanations or descriptions about the process and characteristics of the model service composition for simulation in detail. 2) There is no consideration about the cooperation relationship between the model services in the MSCS problem for simulation requirements. 3) Existing common QoS properties (e.g., Simulation execution time, Simulation Cost, Simulation Result Reliability, etc.) are not suitable for the evaluation of the MSCS problem.

With the consideration of the above issues, the process of model service composition in cloud for simulation application requirements is illustrated in detail. A simulation model service scenario in the medical area considering the cooperation relationship between the model services in MSCS is given. Furthermore, a new intelligent evolutionary algorithm named cooperation-aware NSGA-II algorithm using adaptive crossover and mutation operators (CA-AO-NSGA-II) is developed for solving MSCS issues based on model maturity. In this algorithm, the probabilistic initialization population algorithm is introduced to speed up the convergence of the algorithm. Then, the local search algorithm and adaptive crossover and mutation operators are designed to increase Pareto fronts and improve the quality and convergence speed of the optimal solution. The experimental results demonstrate that the CA-AO-NSGA-II provides better feasible solutions of the model service composition in cloud with lower time consumption for solving the MSCS issue with two objectives, especially as the number of cooperation constraints and subtask increases.

The **key contributions** of this work are as follows:

• This paper introduces model maturity as a metric for QoS evaluation to provide a comprehensive assessment of the whole lifecycle of the simulation model in cloud environments,

• This paper employs the cooperation relationship between model services as a parametric indicator to dynamically calculate the value of the overall maturity of the combined model,

• This paper proposes an improved algorithm based on NSGA-II to solve the MSCS issue for the composition and optimization selection of model services in cloud environments.

The rest of this paper is organized as follows. In Section 2, related works about model composition in simulation, MSCS, and some multi-objective evolutionary algorithms for solving MSCS issue are reviewed briefly. In Section 3, the process of model service composition in cloud, and a medical simulation application scenario are described. Then, the formulation of the multi-objective MSCS is given in Section 4. Furthermore, a novel algorithm termed CA-AO-NSGAII for solving MSCS based on model maturity is proposed in Section 5. Section 6 presents experiment results and analysis to show the feasibility and effectiveness of the proposed method. Finally, conclusions and future works are summarized.

2. Related Work

There are mainly two directions for the model service composition and optimization research, i.e.,

whether the model services can be combined, and whether the combined services can satisfy the expectations. The corresponding methods usually are, a. filtering the model services from the perspective of syntax or semantics, i.e., functional composition. b. Using QoS methods to judge whether the combined effect has achieved the expected demands, i.e., non-functional composition. The related research work will be briefly reviewed from the following three aspects.

2.1 Model Composition in Simulation

The research is mainly carried out from three aspects: model description language, modeling methods for model composition, model composability validation. From a perspective of a description language, Zeigler et al. [1] proposed discrete event system specification (DEVS), and gave the description of the atomic model and coupled model, which are used to build top-down hierarchical structures to construct composable models. Friedenthal et al. [12] provided a practical guide to SysML, a unified system modeling language. Generality and extensibility features make the models built through the language highly composable. From a perspective of modeling methods for model composition, Tolk et al. [13] proposed a conceptual interoperability model from the perspective of simulation model interoperability and divided the interoperability into five levels according to the nature of the exchanged data and the level of standardization of the interface, and then performed hierarchical modeling of model composition. Wittman et al. [14] proposed the product line architecture framework (PLAF), which incorporated the application, product, and component layers, and provided a combining mechanism to combine different product components into composite product components, such as simulation models. In the validation perspective, Pitty et al. [15] proposed semantic composability theory (SCT) to integrate simulation components into new simulation applications. SCT validates if model combinations are accurate from a syntactic and semantic perspective.

In addition, from other perspectives, Kang et al. [16] proposed a composition model based on hierarchy color Petri net (HCPN) to model command and control procedure of surface air defense by reusing existent simulation models. Alpdemir [17] proposed SiMA: a simulation construction environment that supports simulation models' composability through a simulation construction toolchain. Mittal [18] designed DEVS/SOA architecture, which provides the crucial feature of run-time composability of coupled systems using SOA. Pitty et al. [19] designed different software frameworks for model composition, which are intended to simplify assembling a complex model or simulation system to promote the reuse of the component models. Cayirci [7] surveyed the MSaaS architectures and deployment strategies, and gave the differences between MSaaS and software as a service.

From the above research, we can see that from the functional composition aspect, select and match the right models that can be combined usually using methods of semantic composability of model services, interoperability of model services, and validation of model service combinations. However, few studies give the specification on the non-functional requirements of the model composition. They are studied mainly from the perspective of whether the syntax or semantics of models meet the requirement of composability. Moreover, the form of the model composition has also transitioned from an SOA-based architecture to a cloud-based model service composition.

2.2 Model Service Composition in Cloud

Many techniques and methods, such as architectures, middleware, and simulation tools, are used to determine whether or not model services are composable from a functional perspective. Among them, most researches have been done on MSaaS. Taylor et al. [20] presented business models based on CloudMSE platform experiences supporting MSaaS, which can provide simulation services. Wang et al. [21] built an MSaaS middleware called CloudRISE to simplify the management of a variety of M&S resources.

Bocciarelli [22] designed an available MSaaS platform named SOASim based on microservice to achieve a fine-grained combination of model resources. Wainer [23] et al. gave a novel architecture MAMSaaS. Wang [24] also introduced an architecture named SAMSaaS to deploy and compose M&S resources as services, so as to improve model reuse through model composition based on semantic.

In model service composition research, the concept, architecture, and technical implementation of MSaaS are becoming more and more mature. However, there is no specific optimal algorithm for the combination of model services in MSaaS. Most of the literature focuses on connecting and collaborating between model services from a technical or architectural perspective. It does not give much researches on model service composition algorithms from the QoS perspective. Model services are different from manufacturing services and computing services, which have their features and characteristics. They should be studied based on the differences between them, under different constraints, to provide high-quality model services. Considering the above situations, the process and a model for MSCS from the perspective of QoS based on model maturity are conducted in detail in the following sections.

2.3 Service Composition and Optimal-Selection Algorithms

In recent years, cloud service composition has drawn much attention mainly to its architecture (Graph-based, agent-based service composition, WS-BPEL), dynamically composition and optimal-selection algorithms, and services correlation relationships. Some researches and results have been achieved. From the perspective of composition and optimal-selection algorithms, Alrifai et al. [25] proposed a hybrid approach that decomposed the best of global QoS constraints into local constraints with mixed-integer programming to address optimal problems. Chhun et al. [26] presented a QoS ontology with functional and non-functional properties for service selection and reuse. Liang et al. [27] proposed a logistics-involved QoS-aware DRL-based CMfg-SC, and designed a dueling Deep Q-Network (DQN) with prioritized replay named PD-DQN, which demonstrated the effectiveness and advantages of DRL in solving the CMfg-SC issue. From the perspective of services correlation relationships, Wang et al. [28] considered two composability-oriented and quality-oriented correlations, and proposed a many-objective algorithm named HypE-C to solve CASC problem in cloud manufacturing. Deng et al. [29] proposed a novel method of service selection called the cooperation-aware service pruning (CASP) to manage QoS correlations by accounting for all services. Luo et al. [30] proposed a business correlation framework in the ecosystem.

Hence, the solution for optimal selection of service composition is primarily based on QoS evaluation methods and metrics, and the effectiveness of the combination also depends on the selection and appropriate improvements of the evolutionary algorithms.

However, there are no optimal algorithms about solving MSCS issue to address the simulation requirements. Furthermore, few studies specify how the cooperation and constraints affect the feasibility and quality of the model service composition in simulation application fields. Therefore, we leverage a service composition optimization approach to address the problem of MSCS based on model maturity and consider the cooperation relationship between model services to select a more suitable model composition that meets the simulation requirements.

3. The Process of Model Service Composition in Cloud for Simulation and Application Scenario

3.1 The Process of Model Service Composition in Cloud for Simulation

According to the report about MSaaS described by NATO MSG-131 [31], there are various kinds of services for simulation, including modeling services, V&V services, model services, and simulation

application services, and so on. All of these services for simulation are submitted by service suppliers who own many simulation resources. In this paper, we only consider the model services for simulation. Generally, one model service can fulfill the demands of one single simulation task. However, it cannot execute a complex simulation task, which needs several different model services to be combined to complete a complex or a system-level simulation requirement. The more single function the model has, the more efficiently the model runs. Fig.1 shows the process and framework of model service composition in cloud environments, and the cloud service platform contains three main functional modules: Model service certification and storage(C&S) module, Task decomposition, service matching and optimal-selection(D&M&O-S) module, and Model service deployment and execution (DP&E) module.

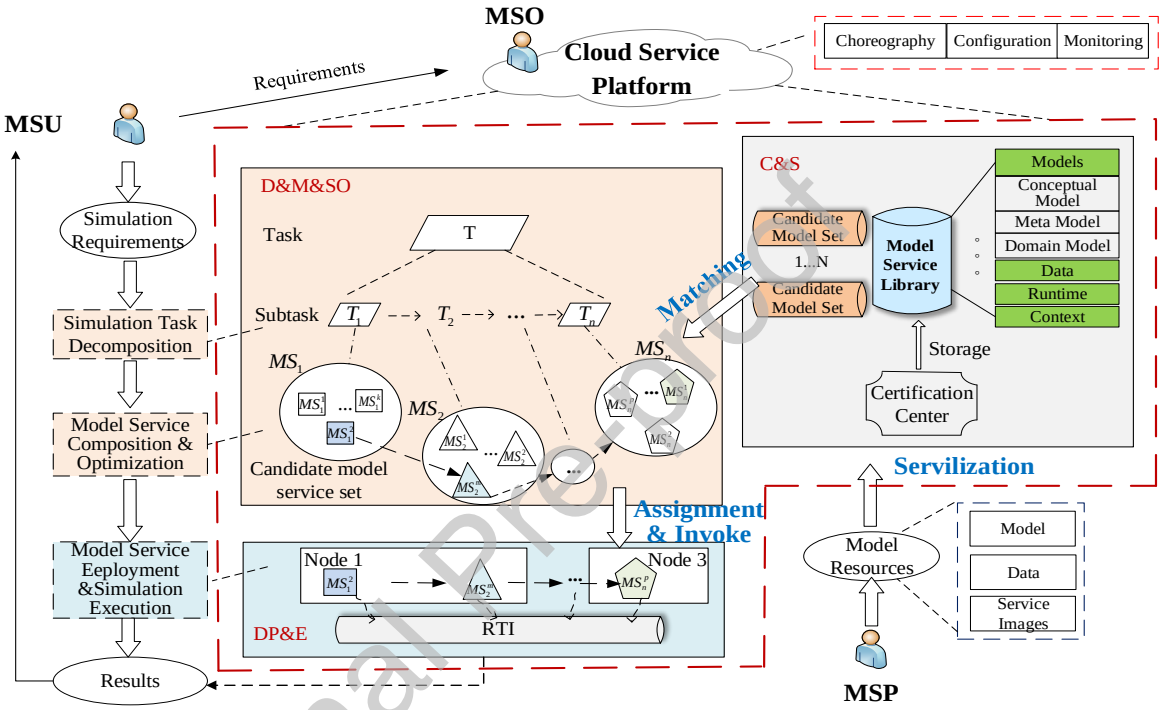

Figure 1. **The process and framework of model service composition in cloud**

- Model Service Certification and Storage(C&S)

The module of model service certification and storage (C&S) performs the authentication and storage of model service resources. After model resource servitization, the models uploaded by MSP need to be certified usually by a third-party authority to check whether it can meet specific quality standards, whether it can perform certain functions, and whether it is credible. After certification, the models will be stored into the model service repository in cloud with different granularity (e.g., conceptual model, meta-model, domain model) or classification of different functions to form model service candidate sets. At the same time, the data, runtime, and contextual information related to the models are also stored.

- Task decomposition, service matching and optimal-selection(D&M&O-S)

Firstly, the task or requirement description submitted by MSU should be decomposed, the decomposition process needs to go through the process of functional requirements analysis, process requirements analysis, and finally form an abstract combination of services, i.e., subtasks $\{T_1, \cdots, T_i, \cdots, T_N\}$. Then, through service matching techniques such as similarity computation, interface/function matching, process matching, and semantic matching, the corresponding candidate sub-services $\{MS_1, \cdots, MS_i, \cdots, MS_N\}$ are selected for each subtask from the candidate service sets stored

in the model library. Finally, selecting the specific model services $\{MS_1^2, \cdots, MS_2^m, \cdots, MS_N^p\}$ that meet the user's demands by a service composition and optimal algorithm based on different optimization objective functions.

- Model service deployment and execution (DP&E)

After forming a model service composition solution or path, the model service needs to be deployed and configured. The deployment optimization algorithm specifies which nodes are assigned to which model services and the relevant operating parameters of the model configured. Each node host is deployed to run several different model service mirrors using container technology to get the final computation result. In addition, before the entire simulation run, model transformation (e.g., PyBPMN-to-UML, SysML-to-HLA, HLA-to-Code, etc.), which is used to generate models as well as the code that implements executable services from abstract models is required, so as to ensure the uniform and regular operation of the entire combined model services. Each model service is deployed on cloud infrastructure or middleware like SOASim [32], CloudRISE [21] to make them better operated and supervised. Finally, the orchestration service is deployed to properly manage the execution of the simulation models with one RTI or some simulators [33]. MSO is responsible for monitoring the operational status of the model services and managing the feedback of the simulation results to the MSU. After getting the results, MSU can provide feedback to the platform.

In addition, we also need to perform VV&A on single or combined models. Generally, dynamic testing methods include spectral analysis methods and feature-based difference verification methods for VV&A. At the model service configuration and execution level, the platform needs to support VV&A activities [34] and other validation methods [35] for each model to ensure accurate interaction between model services and meet the simulation requirements tasks.

In summary, we should try to upload single, fine-grained models to the platform and store them in a reusable model repository. It can minimize the problem of difficult reuse and combination of model services caused by the strong coupling with the simulation environment, the incompatibility between model architectures due to the complexity and versatility of the model services themselves.

3.2 Application Scenario

In order to describe MSCS problem more clearly, an example of simulation service scenario in the healthcare field based on DTH [36] is presented. As shown in Figure 2, a simulation task for healthcare service contains five subtasks, which need to invoke monitoring model service (MMS), examining model service (EMS), diagnostic model service (DMS), resource scheduling model service (RSMS), rehabilitation model service (RMS). Each of these five models corresponds to the main five processes that a complete healthcare service needs to go through, respectively. From the second to the last model service, each service receives input data from the previous service for processing the subtask. Finally, after a complete run, the user gets the final simulation results. The service demanders or users mainly include two kinds of people. One category is patients, who want to get some valuable suggestions through the whole simulation service. The other category is simulation participants, who want to get some simulation data for making some decisions. Each subtask can be selected from the candidate service sets stored in the model library in cloud with different functions.

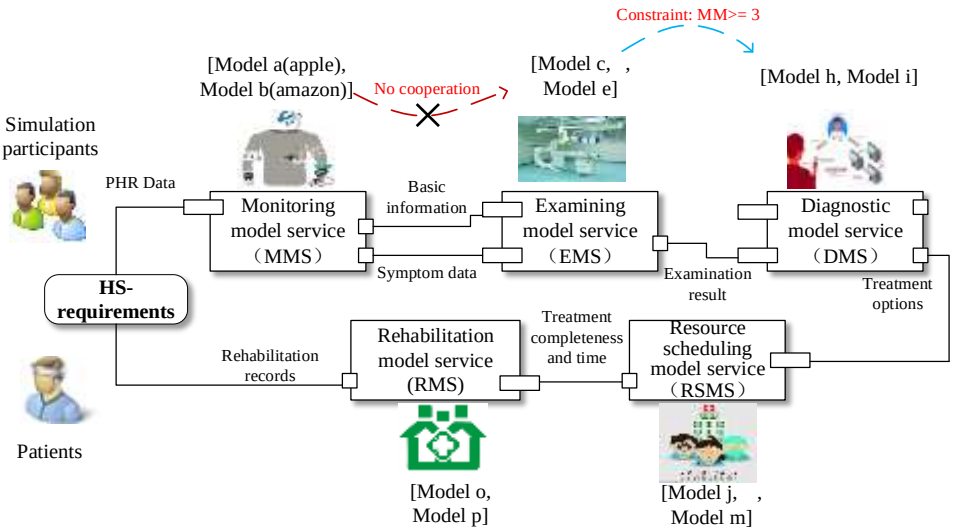

Figure 2. **A scenario of DTH model services composition in precision healthcare**

The purpose of model service composition is to select an appropriate model service from the candidate service sets for each subtask to complete one simulation task to meet the user's requirements when combined and executed together. However, the candidate model services are not independent of each other, except that the interfaces should be identical. They must also satisfy the relevant constraints to combine them and serve their purpose. There are two examples shown in Figure 2. Below, we describe each of the three scenarios in detail.

Firstly, in the candidate model service sets of MMS, if we choose model service b, assuming its provider is Amazon, and it does not cooperate with the model service provider e in EMS due to commercial competition or other factors, so we cannot choose model service b and model service e together at the same time. Secondly, if we choose model service c of EMS, and it requires the following service provider of DMS to provide the model with a model maturity level [9] of Level 3 or higher (e.g., MM>=3); otherwise, the accuracy and quality of the whole task cannot be guaranteed. Therefore, the selection of model services for subtasks should not only consider the static QoS metrics (e.g., time and cost) when they are composed, but also should record some dynamic metrics such as model maturity, multiple constraint relationships, and so on. And judge if they are feasible when they are selected and combined.

In this paper, the above two kinds of relationships are termed multiple constraints, which only occur between two adjacent services and will not occur across services. That means the first service in a sequence of service composition has no constraints unless the user has constraint requirements on it. If there is a constraint conflict between two neighbored services, then they will not be able to be composed to perform a simulation task.

4. Formulation of the Multi-Objective MSCS in Cloud

In this section, the formulation of multi-objective MSCS in cloud from the perspective of QoS is presented. And due to the characteristics of the model services for simulation and the importance of the indicator parameters, two objectives of QoS indicators, namely model maturity (MM) and model resource energy consumption (ME), are selected to build the mathematical model for the MSCS problem. The topology of model service composition usually consists of four types, i.e., sequence, parallel, selective and circular. In this paper, we mainly consider the models of the sequential topology of model service composition.

Table 1: Notations and explanations

Notations	Explanations
T	simulation task submitted by users
T_i	i th subtask of task T
N	total number of subtasks of task T
MS_i	candidate service set for the i th subtask T_i
k_i	The total number of candidate model services in MS_i
MS_i^j	The j th candidate service of MS_i for the subtask T_i
$MM(MS_i^j)$	Vlue of the j th model maturity of the MS_i
$ME(MS_i^j)$	Vlue of the j th energy consumption of the MS_i
CS_m	Alternative service set of MS_m
$IoC_{i,i+1}^{j,p}$	The index of collaborations between service MS_i^j and service MS_{i+1}^p
$NoC_{MS_i^j}^{MS_{i+1}^p}$	The number of the cooperation between service MS_i^j and service MS_{i+1}^p
Mc_i	The cooperation index value of the i th sub-service in the cooperation relationship matrix N*N

Notations and specific explanations of some key terms in MSCS model are given in table 1. Let $T = \{T_1, \ldots, T_i, \ldots, T_N\}$. denotes a simulation task in which N is the total number of the subtasks decomposed by cloud platform, and T_i is the i th $(i = 1, \ldots, i, \ldots, N)$ subtask of T. It is assumed that there are k_i candidate model services available for T_i, and the corresponding candidate service set is $MS_i = \left\{MS_i^1, \ldots, MS_i^2, \ldots MS_i^j, \ldots, MS_i^{k_i}\right\}$, where MS_i^j is the j th candidate service of MS_i for the subtask T_i.Therefore, theoretically, there are a total of $\prod_{i=1}^{N} k_i$ possible compositions without any constraints. Usually, due to the enormous variety and number of services in cloud environments, using a brute force search to select the optimal combination is infeasible.

In this paper, the main QoS properties of the model service we considered are, model maturity (MM) and model resource energy consumption (ME). Why choose the two objectives? There are four reasons for this.

Firstly, for the candidate model sub-services that perform the same subtask with the same function at the same granularity, their cost and time are not very different and cannot effectively select appropriate model sub-services by those metrics for a simulation requirement. Secondly, the model services are verified with the VV&A operations by the certification center and then stored in the model repository to provide candidate model services, so the indicators such as availability, reliability, and credibility do not vary as much as those of cloud manufacturing services. These indicators will not change during implementation and even affect the final QoS evaluation. Thirdly, model maturity is an evaluation indicator of the whole lifecycle of a model service. Considering the multiple constraints, the differences in the overall model maturity exhibited by different combinations of the models can be relatively large. Fourth, due to the development and application of container technology and other New IT technologies, the simulation time to execute the model service will not be much different. However, the resource consumption to execute the model service will be very different, especially when the resources are consumed by different subtasks, which also significantly impacts the whole model composition. In

summary, considering the characteristics of the model service composition, two metrics, model maturity, and model resource consumption, are very suitable to be chosen as QoS assessment metrics for the MSCS issue.

4.1 The optimization objectives and multiple constraints

4.1.1 Model Maturity

Model maturity is another indicator of model quality evaluation, the same as model credibility and model fidelity [35]. It pays much attention to record the status and changes of a model in its whole life cycle. It can have a better evaluation for an evolving model during the whole lifecycle of the model [9]. It measures how well a model meets the expected effects and application goals along with the time and frequency of using the model increase. With the help of an index system for model maturity assessment, we evaluate the process of model construction, model use, and model management, then a combination of qualitative and quantitative methods can be used to give the maturity value of a model at a certain stage. The value range of the model maturity is [0,1]. There are 5 levels to evaluate the state of a model during the whole life cycle named initial level, verified level, reusable level, collaboration level, optimal level, respectively. And the corresponding value ranges are [0,0.15], (0.15,0.3], (0.3,0.15], (0.5,0.75], (0.75,1]. According to this hierarchy, the models are generally at level 3-the reusable level, so all the candidate models in the model repository have a maturity value of at least 0.3.

Model maturity is different from the CMMI model and SaaS maturity model [37]. The CMMI model emphasizes the software development capability of a software development organization, and it is a certification system for the management and R&D (research and development) capability of an enterprise or organization. The object of the assessment is the enterprise or organization. The SaaS maturity model is a metric for software architecture in a cloud environment. It evaluates the software architecture capabilities based on the metrics of configurability, high performance, and scalability to determine whether the design capabilities of the software architecture meet the expected results, and the object of the assessment is the capability of software architecture design in cloud. The object of model maturity assessment is simulation model, and the evolution state of the whole life cycle of the model is studied, including the model construction stage, the model use stage, and the model management stage. The three of them have different assessment targets and different scope and responsibilities of assessment.

4.2 Model Resource Energy Consumption

Model resource energy consumption mainly refers to the computational resource energy consumption of the model service to execute a simulation task using the model. The energy consumption of computing resources comes mainly from the occupation and consumption of CPU and memory, so ME can also be represented as:

$$ME\left(\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}\right) = \mathrm{M}E_{memory}\left(\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}\right)+\mathrm{M}E_{cpu}\left(\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}\right) \quad (1)$$

The energy consumed by each model service is different with each other and multiple different model services could be allocated on one compute node in one host, thus, there is a need to limit the individual energy consumption of each model service, and in this paper, we set the ME of each model service to be less than 140.

4.3 The Multiple Constraints between Model Services

Multiple constraints between model services can have a large impact on selecting candidate model services, and the presence of conditional constraints in the services is consistent with the actual situation,

so the relevant conditional constraints need to be taken into account. This paper mainly considers two kinds of multiple constraint relations, as mentioned in 3.2, which are exclusion constraints and conditional constraints [28]. The first one: the exclusion constraint, which means model services are not combinable, i.e., the model service $\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}$ cannot be combined with the model service $\mathrm{MS}_{\mathrm{i+1}}^{\mathrm{p}}$ due to business conflicts of interest or other restrictions, $(\mathrm{i} = 1, \ldots, \mathrm{i}, \ldots, \mathrm{N})$. The second one: conditional constraint, maturity level requirements, i.e., model services require that the maturity level of the next candidate services must be larger than 2 (or 3), that is, the maturity level must be 3 or 4 (a total of 0 to 4).

Thus, in addition to the last service in the sequence of composite model services, each of the other execution services for subtasks may have 0 to 2 constraints for the following neighboring service. If service $\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}$ for subtask is selected, and any other services in this composite service do not belong to the alternate candidate set CS_m for $\mathrm{MS}_{\mathrm{i}}^{\mathrm{j}}$, then, this model service composition will not be feasible.

4.2 The Objective Function and Constraints of MSCS

4.2.1 Objective Function

Reducing the energy consumption and maintaining a high maturity of the model service composition simultaneously in the cloud is a challenging problem, especially when it goes with multiple constraints, which is a long-term challenge. Also, there are lots of kinds of constraints in model service composition. In this paper, we use a multi-objective optimization algorithm to solve the problem with the two objectives. Thus, MSCS problem is formalized as

$$\max\ f(x) = (f_1(x), f_2(x)) \tag{2}$$

Where $\mathrm{f}_1(x)$ denotes the value of the overall model maturity of the model service composition. $\mathrm{f}_2(x)$ denotes the value of the sum of ME of the model service compositions. And $\mathrm{X} = (x_1, \rightarrow x_i, \rightarrow x_N)$ is the solution vector which represents one possible composite service path, and it must also satisfy the multiple constraints.

In the two objectives, the positive indicator is MM, and the negative indicator is ME. We should convert all indicators into positive indicators so that the larger the value of $f(x)$, the better the overall effect of the model service composition.

One of the efficient ways to solve multi-objective optimization problems is to convert multi-objective problems into single-objective problems and then solve the single-objective problems using the common methods such as Weighted Sum Method, ε-Constraint Method, and Min-Max Approach. In the paper, we use the linear weighting method, i.e., Weighted Sum Method, to convert this multi-objective problem into a single-objective problem, the fitness function $f(x)$ of single-objective is defined as follows:

$$f(x) = \begin{cases} 0 & ,if\ s_i \notin CS_i \\ \sum_{i=1}^{N} \left(\dfrac{w_{f_1} * Nor\left(M(\mathrm{S}_i^j)\right)}{N} + w_{f_2} * Nor\left(\dfrac{1}{E(S_i^j)}\right) \right) & ,else \end{cases} \tag{3}$$

Subject to:

$$0 < \sum_{j=1}^{k_i} ME(MS_i^j) \le ME_{\max}, \forall i = 1, 2, \cdots N,\ j = 1, 2, \cdots k_i \tag{4}$$

$$0 < g\left(MM\left(MS_i^j\right)\right) < 1, \forall i = 1, 2, \cdots N,\ j = 1, 2, \cdots k_i \tag{5}$$

$$0.3 < MM\left(MS_i^j\right) \le 1, \forall i = 1, 2, \cdots N,\ j = 1, 2, \cdots k_i \tag{6}$$

$$0 < ME\left(MS_i^j\right) \le 140, \forall i = 1, 2, \cdots N,\ j = 1, 2, \cdots k_i \tag{7}$$

Where w_{f_1} and w_{f_2} are the weights of MM and ME, respectively. And $w_{f_1} + w_{f_2} = 1$. CS_i is the alternative model service set of T_i with multiple constraints.

If the composite service is infeasible, i.e., $MS_i \in CS_i$, its fitness values of the two objectives are set as 0. Among them, N is the number of subtasks for one simulation task. Function $g\left(MM\left(MS_i^j\right)\right)$ in formula (5) represents the value of the overall maturity of the composed model services. Formula (5) and (6) imply that the model maturity value ranges in [0,1], and the value of MM of one single candidate model service must be larger than 0.3. And formula (6) means that ME of one model service should not exceed the maximum value.

Due to the diversity of QoS assessment metrics, the units of each metric are different and need to be unified to the same interval unit to facilitate our research. In this paper, we use the critical value method to normalize the data, the formalization as follows.

$$Nor(\bullet) = \begin{cases} \dfrac{q_{ij} - \min(q_{ij})}{\max(q_{ij}) - \min(q_{ij})}, & if\ q_{ij} \in ST^+ \\ \dfrac{\max(q_{ij}) - q_{ij}}{\max(q_{ij}) - \min(q_{ij})}, & if\ q_{ij} \in ST^- \end{cases} \tag{8}$$

$q_{ij} \in ST^+$ denotes q is a positive factor set including MM, while $q_{ij} \in ST^-$ denotes q is a negative factor set which involves ME and other domain attributes for simulation.

Thus, the fitness function of the two objectives is as follows, respectively.

$$f_1(x) = \frac{\sum_{i=1}^{N} Nor\left(MM\left(\mathrm{ss}_i^j\right)\right)}{N} \tag{9}$$

where $\mathrm{f}_1(x)$ denotes the value of the overall model maturity after the effective composition of model services for one simulation task.

$$f_2(x) = \sum_{i=1}^{N} Nor\left(\frac{1}{MM\left(\mathrm{ss}_i^j\right)}\right) \tag{10}$$

where $\mathrm{f}_2(x)$ denotes the value of the overall model energy consumption after the effective composition of model services for one simulation task.

There are many ways to calculate the MM value of the overall composable model, Eq. 5 is only one of them. And the following describes a way to calculate the weights of MM that considers the service cooperation of each model service for subtasks, which makes the value of the overall model maturity closer to the actual situation.

4.2.2 Variable-Weight Objective Function of MM

Model service composition is a series of model services assembled in accordance with specific processes and rules to work together to complete a simulation task, so the existence of some model service

relationships will have an impact on the whole process of service composition, especially on the quality of the entire model service composition (composition reliability, composition model maturity, success rate, etc.). In this paper, the purpose of the MSCS is to choose the optimal solution for the model service composition under the sequential structure. It requires that the overall maturity of the composed model should be high, so determining the overall maturity of the composed model services by the maturity value of the individual models is also an important issue worth investigating. We give a way to get the solution of overall maturity of the composed model services as other QoS properties as shown in formulation 11, i.e., the weighted average method. However, it might be inaccurate to use this approach to solve MSCS issue due to different cooperation situations between model services. Consequently, we introduced a metric called the cooperation index (i.e., $IoC_{i,i+1}^{j,p}$ in table 1) to indicate the number of times and the effectiveness of cooperation between two adjacent services, indicating the strength of the correlation between the services and the impact of this group of services on the overall service composition.

The weight value of each model service for subtasks in each feasible solution consists of two parts. One is mean weight w_{aw} in formulation 12, and the other is correlation weight w_{cw} shown in formulation 13, and then the weight value is calculated by formulation 11.

$$w_M\left(MS_i^j\right) = \frac{w_{aw}(MS_i^j) + w_{cw}(MS_i^j)}{2} \tag{11}$$

$$w_{aw}(MS_i^j) = \frac{1}{N} \tag{12}$$

$$w_{cw}(MS_i^j) = \frac{Mc_i}{\sum_{i=1}^{N} Mc_i} \tag{13}$$

$$\begin{bmatrix} 1 & 0.5 & 0 & 0 & 0 \\ 0.5 & 1 & 0.2 & 0 & 0 \\ 0 & 0.2 & 1 & 0.7 & 0 \\ 0 & 0 & 0.7 & 1 & 0.4 \\ 0 & 0 & 0 & 0.4 & 1 \end{bmatrix} \tag{14}$$

Suppose the matrix of cooperation index between model services in a set of feasible solutions is shown in formula 13, then the Mc_i value for each model service is the sum of the values of row i or column i, i.e., 1.5, 1.7, 1.9, 2.1,1.4. Then, the weights of each model in this combined solution are 0.17, 0.20, 0.22, 0.25, 0.16 according to Eq. 11, 12, 13.

5. Multi-Objective Algorithm CA-AO-NSGA-II

Evolutionary algorithms have undergone three main generations of research. The first generation of evolutionary multi-objective optimization algorithms is mainly based on non-dominated ranking and small habitat techniques to solve multi-objective optimization problems. The representative algorithms are MOGA, NSGA and NPGA. The second generation of evolutionary multi-objective optimization algorithms is marked by elite retention strategy. Many excellent algorithms have been born, such as SPEA, SPEA2, PAES, PESA-II, NSGA-II and so on. The third generation is dominated by the study of high-dimensional multi-objective optimization, such as MOEA/D-DE, NSGA-III, HypE, and other

algorithms. Since only the two multi-objectives are involved in this paper, the classical algorithm NSGA-II performs very well when there are fewer multi-objectives. On the contrary, some high-dimensional multi-objective algorithms do not perform as good as the NSGA-II algorithm on low-dimensional objectives. Therefore, we try to use the NSGA-II algorithm and improve NSGA-II algorithm to solve the model maturity-based MSCS problem.

5.1 A Brief Introduction of NSGAII

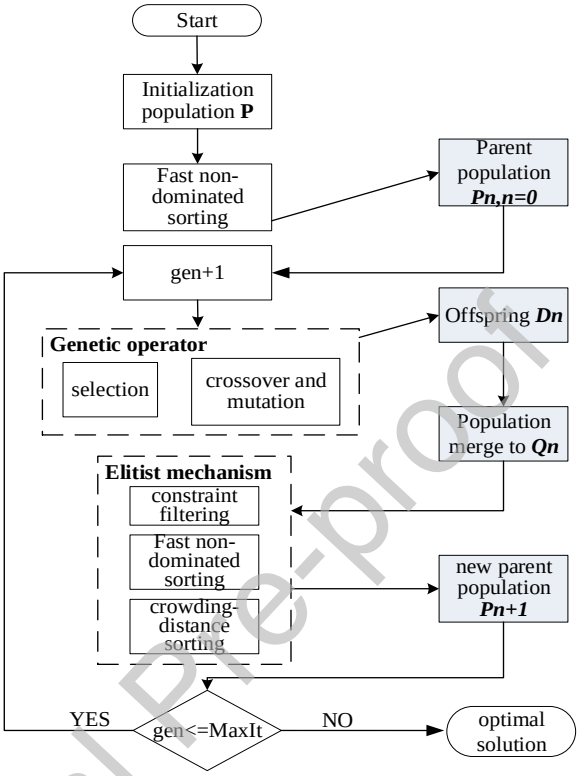

Figure 3. **A flowchart of NSGAII algorithm**

As a second-generation evolutionary algorithm, NSGAII algorithm is one of the most cited classical algorithms in the field of evolutionary multi-objective optimization by SCI [38]. It reduces the computational complexity of the problem by introducing three algorithms: fast non-dominant sorting, elite strategy, and crowding distance sorting, as shown in Figure 3, while making the individuals in the Pareto fronts as uniformly distributed as possible and maintaining good individuals, and improve the overall evolutionary level of the population. This paper only considers two objectives, i.e., MM and ME, and it can efficiently achieve better optimization results by using NSGAII algorithm. In addition, the cooperation relations between model services must also be considered. Therefore, an improved algorithm called CA-AO-NSGAII is proposed to optimize goals better and solve the MSCS issue efficiently.

5.2 The Proposed Algorithm CA-AO-NSGAII

The core thrust of evolutionary algorithm improvement is the balance of local search and global search. Taking the improvement of genetic algorithm (GA) as an example, the improvement mainly includes three major types of methods: 1) adaptive methods, the strategies for adaptive control improvement of parameters of the algorithm. 2) local search methods, including local search strategies and immune heuristic strategies etc. 3) global search methods, including chaotic variation strategies and small habitat strategies, improving global search ability and convergence speed. Currently, there are many studies in the literature on evolutionary algorithms that focus on improving these three points. In this paper,

we improved the traditional algorithm (NGSA-II algorithm) in three main aspects: 1) the inclusion of adaptive parameters, we added the adaptive method of crossover and variational operators to jump out the local optimums quickly; 2) the global search strategy, we initialized the population selection based on probability; 3) the local search strategy, we used the fold-and-half local search algorithm to find the local optimal for the population. These improvements can be applied to other composition optimization problems as well.

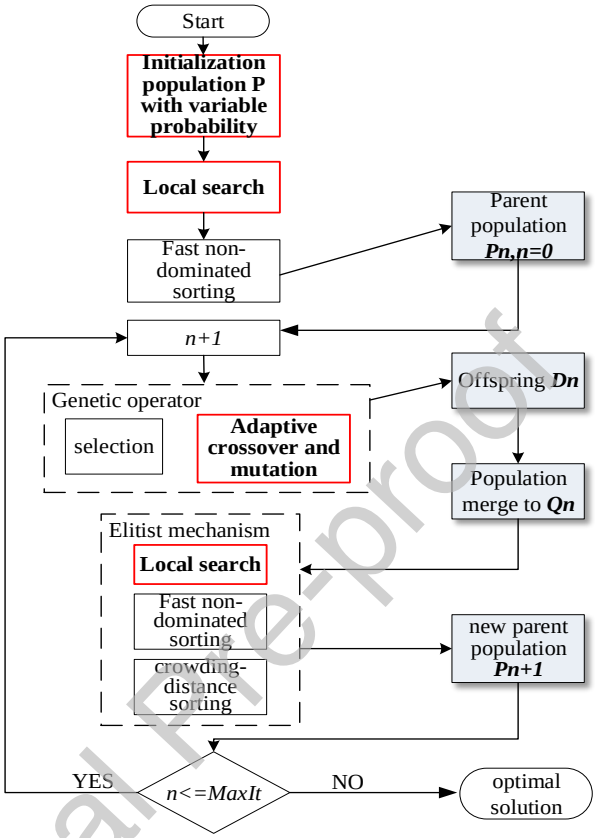

Figure 4. **A flowchart of the proposed algorithm CA-AO-NSGAII**

The flowchart of CA-AO-NSGAII is presented in Figure 4. Firstly, to ensure that the population evolves in a good direction, improve the quality of the population solution, and increase the evolution speed, a population initialization method based on variable probability is proposed to replace the random population initialization method. Secondly, before sorting the initial population P, a local search algorithm is used to make the initial parent populations feasible solutions that satisfy the multiple constraints of model services. This improvement aims to accelerate the convergence speed, and the local search algorithm is also used in the process of elitist mechanism. Thirdly, the crossover and mutation operators need to be dynamically adjusted according to the differences in the individual performance of the population so that the evolutionary process of the population will not be in a stagnant state and generate a competitive offspring Q_n, enabling the algorithm to jump out of the local optimal solution to obtain the global optimal solution. Therefore, the adaptive crossover-mutation operator based on the cooperation index is proposed. These three improvements will be described in detail in the following sections.

5.2.1 The Encoding Method

Considering the complexity of the relationship between adjacent services in the MSCS problem and the dynamics of the number of model services in the candidate model service sets, we use the integer coding method to design chromosome genes, i.e., X_i is a positive integer and the value of it ranges in [1, k_i]. The serial number of X_i represents the serial number of model services in the candidate service sets,

as shown in Figure 5.

$$MS_i^j = X_i, and\ 1 \le X_i \le k_i \tag{15}$$

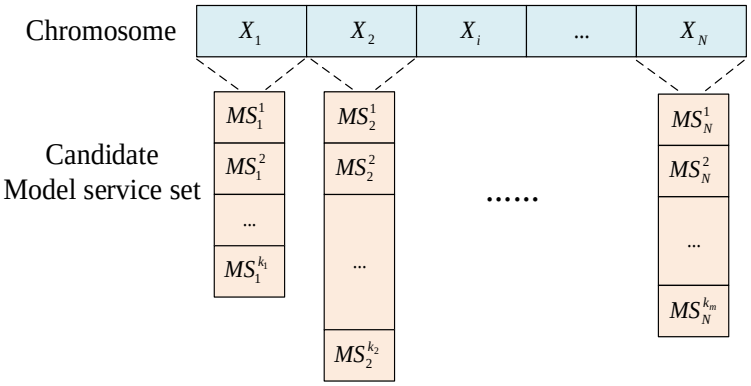

Figure 5**. The encoding method of the proposed algorithm CA-AO-NSGAII**

Taking an example, we set the number of the subtasks as 5, and the number vector of candidate model services of each subtask is [2,3,2,4,2]. One of the chromosomes encoding for a feasible solution of MSCS is [2,1,2,1,1], i.e., the feasible solution x of model service composition is $\{MS_1^2, MS_2^1, MS_3^2, MS_4^1, MS_5^1,\}$.

5.2.2 The Probabilistic Initialization Population Algorithm

Generally, according to the constraint relationship between adjacent services, the selection of the first model service has a great impact on the selection of subsequent services. The more the first service constraints on the next service, the smaller the feasible solution space is. And the entire service composition has no constraints on the candidate services of the first subtask. Therefore, in the population initialization process, try to select the candidate service of the first subtask without multiple constraints so that the entire search space for feasible solutions will be relatively large and much more. The brief flow of the algorithm is as follows.

Step 1. Import the data in the cooperation relationship mapping table of the candidate service set of the first subtask.

Step 2. Calculate the proportion P_{uc} of services without any constraints in the candidate service set of the first subtask.

Step 3. Use P_{uc} as the probability of selecting the first unconstrained gene in the initial population.

Step 4. Randomly selecting the numbers of the other gene positions.

Step 5. Output the initial population.

5.2.3 The Fold-and-Half Local Search Algorithm

The purpose of designing this algorithm is to reduce the problem of non-directional population evolution and slow convergence speed caused by large search space and many multiple constraints between model services. It is to perform a local search and adjustment of the parent population P_n, so that each individual in the population P_n can satisfy the multiple constraints, and then participate into the evolution of the population, thereby avoiding more evolutionary time and avoiding falling into the local optimum. The algorithm steps are described in algorithm 1 in detail with pseudo code. Suppose there is a

model service MS_i^j belongs to individual $MS_i = \{MS_1, MS_2, \ldots MS_i, \ldots MS_N\}$ in population P_n, it has some constraints with the next service. Firstly, starting from the first half of the individual MS_i, determine whether the services are composable, and if not, replace the first half of the model service MS_i with a random model service that has no constraints. Secondly, starting from the second half of the individual MS_i, determine whether the services are composable, and if not, replace model service $MS_{i+1}$with a random model service that satisfies all the multiple constraints of MS_i. Finally, until all individuals are composable, the loop ends.

Algorithm 1: Fold-and-half local search algorithm	
1:	For each composite service solution $MS = \{MS_1, MS_2, \ldots MS_i, \ldots MS_N\}$ in P_n
2:	$P' \leftarrow P_n$ // initialize ***P'***
3:	…***loop***
4:	…do forward algorithm, yielding replace in $MS = \{MS_1, MS_2, \ldots MS_i, \ldots MS_N\}$
5:	……***for*** $z \leftarrow 1\ to\ (N-1)/2$ ***do***
6:	… … …***if*** $CS_{S_{j,i+1}} \neq \varnothing \wedge MS_{i+1} \notin CS_{MS_{j,i+1}}, i \in [1, z]$ ***then***
7:	… … … …update to $MS_i : MS_i \leftarrow \text{random}(As_i)$, where $As_i \in \left\{ CS_m \middle| CS_{MS_{j,i+1}} = \varnothing \right\}$
8:	… … …***end if***
9:	……***for*** $z \leftarrow (N-1)/2\ to\ N$ ***do***
10:	… … …***if*** $CS_{MS_{j,i+1}} \neq \varnothing \wedge MS_{i+1} \notin CS_{MS_{j,i+1}}, i \in [1, z]$ ***then***
11:	… … … …update to $MS_{i+1} : MS_{i+1} \leftarrow \text{random}(As_i)$, where $As_i \in \left\{ CS_m \middle| CS_{MS_{j,i+1}} \right\}$
12:	… … …***end if***
13:	……***end for***
	…***end loop***
14:	$P' \leftarrow clearall(P')$ // clear array ***P'***

5.2.4 The Adaptive Crossover and Mutation Operators

The uniform crossover operator (UCO) is performed to cross the individual gene. In the UCO, the genes of the offspring are randomly obtained from the parent with equal probability, two parents are denoted as parent 1 and parent 2, and two offspring are denoted as offspring 1 and offspring 2 as shown in the Figure 6.

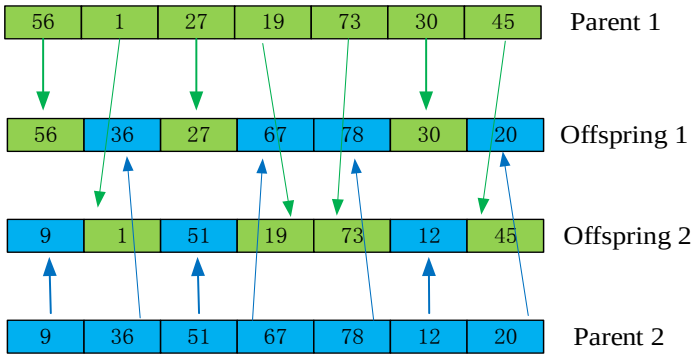

Figure 6. **The uniform crossover operator for CA-AO-NSGAII**

Fig. 7 shows the process of the simple mutation operator. It performs mutation on the values of one or a few randomly designated genes in parent 1 according to the mutation probability, and the value of gene in offspring 1 after mutation must be in the range of [1, k_i]. To ensure the feasibility of the offspring, the gene value of the new individuals should be checked through a local search algorithm.

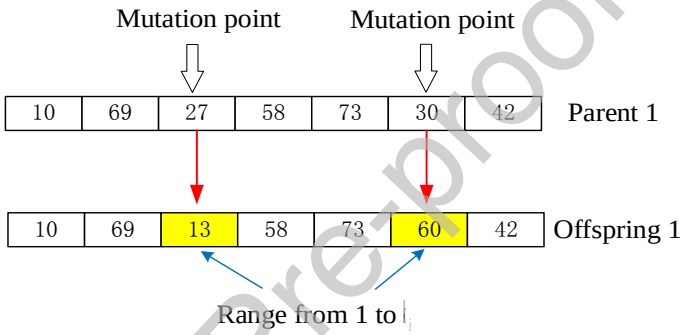

Figure 7. **The uniform mutation operator for CA-AO-NSGAII**

When the fitness value is lower than the average fitness value, the individual is a poor performer, and a large cross and variation rate should be applied to the group. Suppose the adaptation degree is higher than the average adaptation degree. In that case, it means that the individual is a good performer, and a lower cross and variation rate should be applied to it according to its fitness value. In this paper, we improve the adaptive crossover and mutation probability by adaptively adjusting the crossover mutation probability according to the fitness value, and then adding the parameter of the average value of the cooperation index between model services, the higher the cooperation index of the feasible solution, the lower the probability of their crossover and mutation. In this way, evolution will not be in a stagnant state, which allows the solutions to jump out of the local optimal solution to obtain the global optimal solution.

$$P_c = \begin{cases} P_{cMax} * \dfrac{1}{sum\_c / N + e^{\frac{f - f_{avg}}{f_{\max} - f_{avg}}}}, & f \geq f_{avg} \\ P_{cMax}, & f < f_{avg} \end{cases} \tag{16}$$

$$P_m = \begin{cases} P_{mMax} * \dfrac{1}{sum\_c / N + e^{\frac{f - f_{avg}}{f_{\max} - f_{avg}}}}, & f \geq f_{avg} \\ P_{mMax}, & f < f_{avg} \end{cases} \quad (17)$$

As shown in formulas (16) and (17), sum_C represents the sum of the cooperation index of the neighboring model services, and f is the fitness value of the individual MS in the population. If $P_c \leq 0.5$, then $P_c = 0.5$. And we set $P_{cMax} = 0.95$, $P_{mMax} = 0.15$.

6. Experiments and Analysis

In order to verify the effectiveness and feasibility of the proposed method CA-AO-NSGAII for MSCS issues as in the case of Section 3.2, three experiments about the multiple constraints and the algorithm are performed on a PC with Intel core i7+ 2.0GHz, 8GB RAM, Windows 10, and MATLAB R2016a. For all experiments, we set population number equals to 100, and w_{f_1}=0.6, w_{f_2}=0.4. The first experiment is conducted to verify the impact of multiple constraints between model services with three different ratios of constraints. The second experiment compares the performance of CA-AO-NSGAII algorithm with two other classical algorithms, namely, Strength Pareto Evolutionary Algorithm 2 (SPEA2), Non-Dominated Sorting Genetic Algorithm II (NSGA-II). The third one is conducted to show that using different objective functions of the MM, different optimal solutions will be obtained, so that we can enable backpropagation using the new algorithm to compare which objective function of MM yielded a better accuracy of the overall model maturity.

6.1 Experiment I: Impact of Multiple Constraints between Model Services

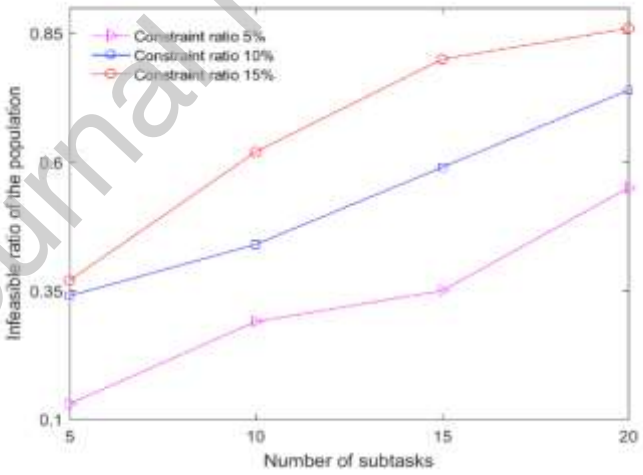

Figure 8. **The infeasible ratio of the population with different constraint ratio**

As discussed in Section 4, each model service may have 0-2 constraints with the next neighboring service, so it is worth exploring how the proportion of different constraints affects the evolutionary selection of the entire population. In all the experiments, we set the number of candidate model service sets range in [0,80], which means that each subtask has a minimum of 0 candidate services and a maximum of 80 services to choose. The proportion of multiple constraints is the proportion of each type of the relationship in the set of candidate model services for each subtask. As shown in Figure 8, the horizontal coordinate represents the number of different subtasks. The vertical coordinate represents the ratio of infeasible solutions in a random population, which is 100.

There are three cases with different constraint ratios tested (i.e., 5%, 10%, 15%), as shown in Figure 8, as the number of subtasks increases, the proportion of infeasible solutions becomes higher and higher. At the same time, as the constraint ratio increases, there are more infeasible solutions for each type of subtasks, maximum of the ratio is up to 86%. Therefore, it is evident that the multiple constraints of model services greatly influence the selection of feasible solutions. In order to ensure the optimal solution of Pareto, the multiple constraints for composability cannot be ignored.

6.2 Experiment II: Comparisons with other Algorithms

In this section, we will evaluate the performance of 3 algorithms in three ways: 1) hypervolume (HV) indicator values of the three algorithms with the different number of subtasks, 2) Maximum objective values obtained by the three algorithms, and 3) the average time consumption of the three algorithms.

The parameter values and some operators of SPEA2, NSGA-II, CA-AO-NSGA-II in the comparison experiments are listed in Table 2. The crossover and mutation operators in the algorithm of NSGA-II, CA-AO-NSGA-II are the same as described in section 5.2.4.

Table 2: Notations and explanations Parameters and operators of the three algorithms

Algorithm	Parameters	Value
SPEA2	Crossover rate	0.95
	Mutation rate	0.15
	Selection operator	Binary Tournament
	External archive set size	100
	Constraint filtering	0
NSGA-II	Crossover rate	0.95
	Mutation rate	0.15
	Constraint filtering	0
CA-AO-NSGA-II	Crossover rate	0.95
	Mutation rate	0.15

The formula for the HV indicator in the case of two objective functions is shown in Eq. 18. First, we sort the Pareto frontier solutions in ascending order according to the value of the first objective function and then calculate HV value according to Eq. 18. The above equation S denotes the set of Pareto frontier solutions, nPop is the number of initial populations. $obj_i(p_j)$ denotes the value of the j th point of p_j on the i th objective in the set, where $obj_2(p_0)$ is set initially$obj_2(ref)$, and ref denotes the position of the reference point, in this paper, we set reference point values equals to [0.3, 0.1*N] for four different scales/subtasks.

$$HV(S) = \sum_{i=1}^{nPop} \left| obj_2(p_i) - obj_2(ref) \right| \cdot \left| obj_1(p_i) - obj_1(p_{i+1}) \right| \tag{18}$$

To compare the performance of the three algorithms on MSCS issues with different number of subtasks, we set the multiple constraint ratio at 5%, and the number of subtasks N takes 5, 10, 15, and 20 with 800 iterations of evolution for each computation. Each subtask has a set of candidate model services, of which the number ranges in [0,80]. Boxplots of the HV values in 30 runs for the above three algorithms are shown in Figure 9 and Figure 10. The Figures show that the HV boxplots of CA-AO-NSGA-II are

obviously higher than the other two algorithms with four different scales/subtasks under 5% and 15% constraints. And as the number of tasks increases, the HV values of the three algorithms increase due to the initial value of the reference point. The result shows that CA-AO-NSGA-II get Pareto optimal solutions with better convergence and diversity for the MSCS problems with different subtask numbers.

Table 3 shows the maximum values of the two objectives obtained by objective functions with different algorithms under the multiple constraint ratio of 5%, and 15%. '+', '=', '-' represent that the CA-AO-NSGA-II algorithm is superior, similar, and inferior to other algorithms, respectively. '5%_5_800' means that the multiple constraint ratio is 5%, the number of subtasks is 5, and the iteration times is 800. In these 16 sets of experiment data, SPEA2 and NSGA-II have only 1 set of data be higher than the new algorithm, SPEA2 has 3 sets of data that are close to CA-AO-NSGA-II algorithm, NSGA-II has 4 sets of data that are close to CA-AO-NSGA-II algorithm. And for most of the remaining data, CA-AO-NSGA-II performs better.

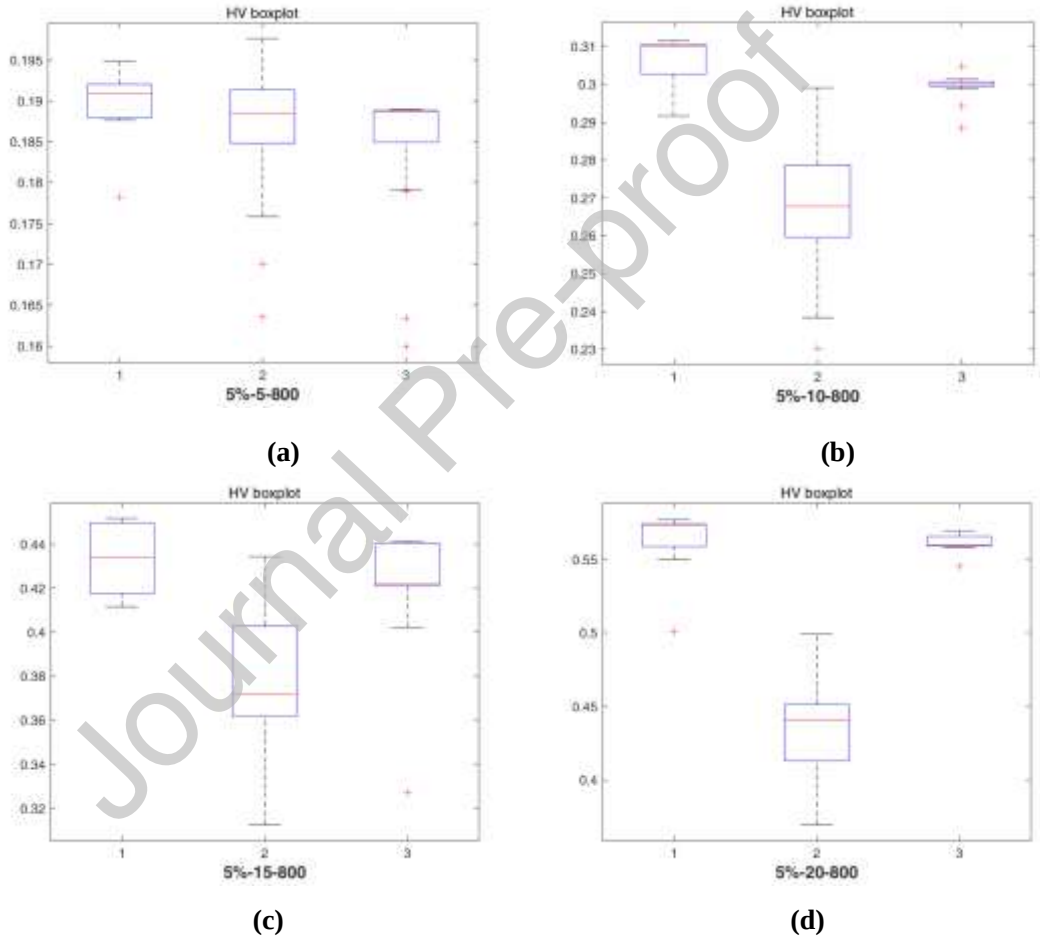

Figure 9. **An example road network and the data structure in the GPU memory HV values boxplot of 3 algorithms with 5% multiple constraints. (a) HV values under 5 subtasks. (b) HV values under 10 subtasks. (c) HV values under 15 subtasks (d) HV values under 20 subtasks. 1-CA-AO-NSGA-II, 2-SPEA2, 3-NSGA-II.**

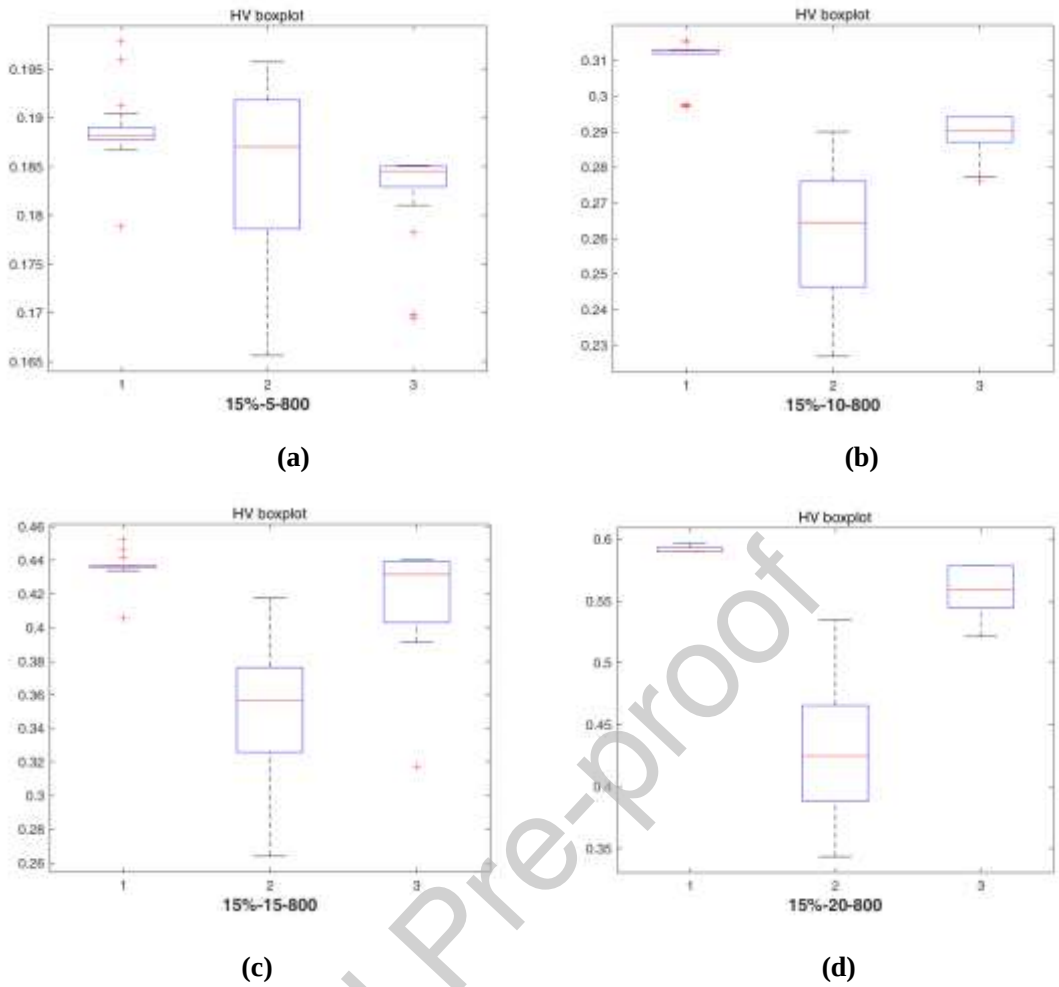

Figure 10**. An example road network and the data structure in the GPU memory HV values boxplot of 3 algorithms with 15% multiple constraints. (a) HV values under 5 subtasks. (b) HV values under 10 subtasks. (c) HV values under 15 subtasks (d) HV values under 20 subtasks. 1-CA-AO-NSGA-II, 2-SPEA2, 3-NSGA-II.**

Table 3: Comparison of the maximum values of different objective functions under three algorithms

Scale	**CA-AO-NSGA-II**		**NSGA-II**		**SPEA2**	
	f_1	f_2	f_1	f_2	f_1	f_2
5%_5_500	**0.5961**	**1.1914**	**0.5975=**	1.1801-	**0.5932=**	1.1842-
5%_10_500	**0.5895**	**2.1295**	0.5784-	2.1144-	0.5767-	**2.1298=**
5%_15_500	0.5695	**3.1597**	**0.5748+**	3.1518-	0.5596-	3.1303-
5%_20_500	**0.5729**	**4.1910**	0.5687-	**4.1888=**	0.5427-	4.1425-
15%_5_500	0.5926	**1.1934**	0.5869-	1.1884-	**0.6000+**	1.1842-
15%_10_500	**0.5852**	**2.1284**	**0.5809=**	2.1233-	0.5602-	2.1161-
15%_15_500	**0.5731**	**3.1520**	0.5626-	**3.1503=**	**0.5731=**	3.1249-

15%_20_500	**0.5736**	**4.1979**	0.5669-	4.1757-	0.5438-	4.1159-
	+/=/-	+/=/-	1/2/5	0/2/6	1/2/5	0/1/7

From the perspective of time consumption, the average time consumed by the CA-AO-NSGA-II algorithm is less than that of NSGA-II, and is little more than that of SPEA2 due to consumption by the more recurrent local searches, as shown in Figure 11. However, in terms of overall performance, the new algorithm performs better than the other two algorithms for addressing the MSCS issue.

From the above Figures, it can be seen that when a local search algorithm is used in the elitist strategy, it helps to search for feasible solutions quickly and improve the convergence speed, saving lots of time. Moreover, with the addition of the adaptive crossover and mutation operators, it is better for us to preserve the optimal solutions and quickly jump out of the local optimum. The convergence and stability of the new algorithm is higher than the other two algorithms.

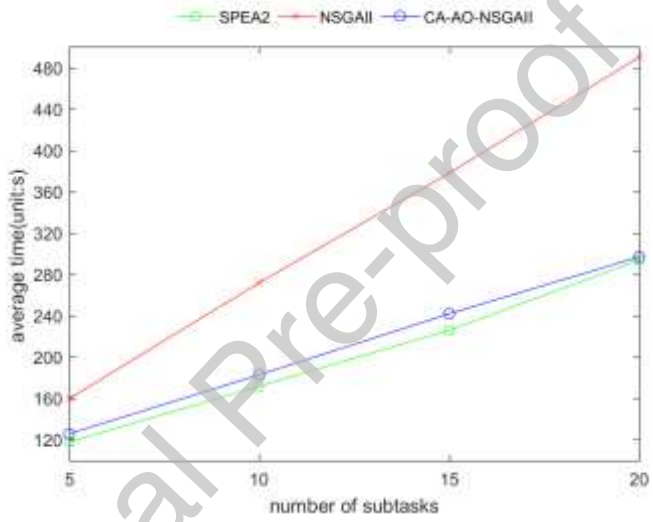

Figure 11**. Average time consumptions of 3 algorithms in 200 iterations under 4 different subtasks.**

6.3 Experiment III: Comparisons of the Solutions Obtained by CA-AO-NSGA-II Using Different Weighting Methods with 5 Subtasks

Table 4: Optimal solutions with different weight methods

Scale	Weight method	Standard deviation	Optimal combination solution
5%_5_800	1	0.0278	49-64-36-42-17
	2	0.0236	49-64-36-42-17
15%_5_800	1	0.0273	49-2-36-42-17
	2	0.0182	49-64-40-42-17

In this section, two different weighting methods are used to get the feasible solutions with the new algorithm CA-AO-NSGA-II, as shown in table4. Method 1 uses Equation 1 and Method 2 uses Equation 2. In order to save space, we give only feasible solutions for the 5 subtasks with 5% and 15% multiple

constraints. From the table, we can see that when the multiple constraints ratio is 5%, the feasible solutions obtained by both weight functions are the same. The standard deviation of the fitness values obtained using the method 2 and Eq. 3 is slightly smaller. And when the constraints ratio comes to 15%, the feasible solutions obtained with the two methods are different. The cooperation index between the services corresponding to feasible solution [49-2-36-42-17] is [0.6,0.3,0,0.4] and the overall value of model maturity is 0.5486. And cooperation index between the services corresponding to feasible solution [49-64-40-42-17] is [0.8,0.8,0.5,0.4] and the overall value of model maturity is 0.5693.

Moreover, the standard deviation of the fitness value obtained by weight method 2 is much smaller than the standard deviation obtained by weight method 1. Therefore, the results show that the correlation between services is affected by the cooperation index. The feasible solution obtained using method 2 is more realistic when the fitness values are closer.

6.4 Experiment IV: A Case Study of Model Service Composition in Healthcare Field

Refer to the scenario in subsection 3.2, this paper gives a case study of a medical simulation model service composition. After the medical emergency simulation task about personal monitoring and warning is submitted to the cloud medical platform, the model service middleware in the service management module in Figure 1 is responsible for completing the simulation task parsing, service search, and matching, etc. It is responsible for selecting the set of candidate model services that satisfy several subtasks of the simulation task. The number of services in the set of candidate model services for each subtask is different. In this paper, the upper limit of the number of candidate services is set to 80, and the candidate services for each subtask are shown in Table 5.

Table 5: Candidate services corresponding to each subtask

Subtasks	Candidate services name	Candidate services sets
T1	Monitoring Model Service (MMS)	$S_1^1,\ S_1^2, \ldots, S_1^{79}\ S_1^{80}$
T2	Examining Model Service (EMS)	$S_2^1,\ S_2^2, \ldots, S_2^{74}\ S_2^{75}$
T3	Diagnostic Model Service (DMS)	$S_3^1,\ S_3^2, \ldots, S_3^{77}\ S_3^{78}$
T4	Resource Scheduling Model Service (RSMS)	$S_4^1,\ S_4^2, \ldots, S_4^{62}\ S_4^{63}$
T5	Rehabilitation Model Service (RMS)	$S_5^1,\ S_5^2, \ldots, S_5^{79}\ S_5^{80}$

Through the digital twin medical model simulation, different model service composition paths for the same model service under different algorithms can be obtained, as shown in Figure 12. From the Figure, it can be seen that due to the limited amount of model service in the experiment, the solutions obtained by three algorithms do not differ much and are the same composition solutions that all can satisfy the optimal solution. However, at the same time, it can also be seen that different algorithms get different service compositions for the model service provided by the same simulation task under the same request. The model maturity for each subtask under the CA-AO-NSGA-II algorithm is [0.95, 0.89, 0.92, 0.99, 0.94], and the overall value of the combined model maturity is 0.5961. The model maturity of each subtask under NSGA-II algorithm is [0.95, 0.89, 0.90, 0.99, 0.94], and the overall value of combined model maturity is 0.5869. The model maturity of each subtask under SPEA-2 algorithm is [0.95, 0.89, 0.90, 0.90, 0.94]. The overall value of combined model maturity is 0.5911. It can be seen that the new algorithm obtains slightly better results for the model service composition, and the overall maturity value of the combined model is also slightly higher.

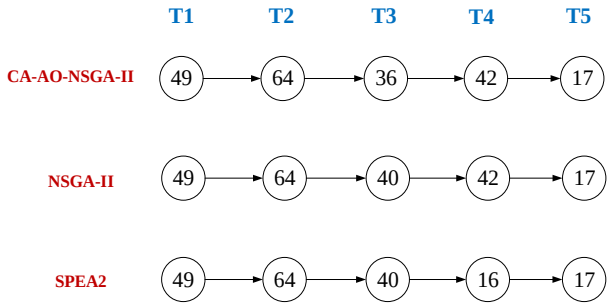

Figure 12**. Model service composition solutions under three different algorithms**

In addition, Figure 13 represents the comparison of the worst, best, and average values of the overall fitness after the combination of model services obtained under the three algorithms for the case of five subtasks. In Figure 13, it can be seen that the new algorithm CA-AO-NSGA-II slightly outperforms the other two classical algorithms both in terms of individual values and mean values. It also illustrates the effectiveness of the improved effect of the new algorithm proposed in this paper in solving the model service combination problem based on model maturity.

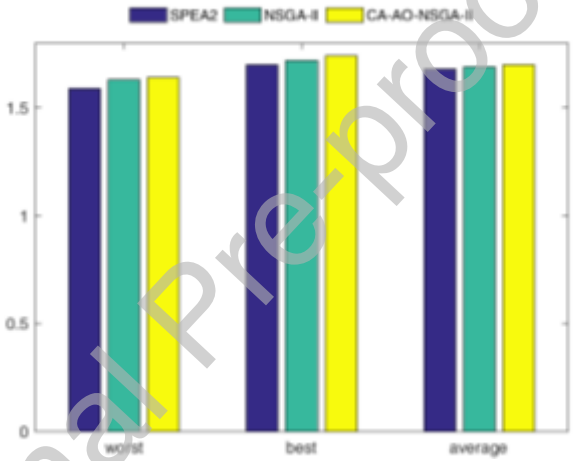

Figure 13**. The comparison of the overall fitness values under three different algorithms**

7. Conclusions and Future Work

With large-scale M&S applications running in cloud environments, highly efficient and accurate methods are desired strongly to solve MSCS issue. In this paper, a novel evolutionary algorithm, named CA-AO-NSGA-II, for addressing MSCS issue in cloud environments was proposed. Considering the large solution space and complexity of multiple constraints between model services, the new adaptive crossover and mutation operators and local search algorithm were designed for more and higher-quality Pareto front solutions. For improving the operational efficiency of the algorithm, probability-based methods for initializing populations were also used in the CA-AO-NSGA-II. Compared with the other three traditional evolutionary algorithms, the average fitness value and time efficiency of CA-AO-NSGA-II were better in solving MSCS issue, especially with the subtask number and constraints ratio increase. In addition, the selection of variable objective functions based on model maturity also provided a new idea to evaluate the quality of the model service composition.

Unlike other service compositions (manufacturing services, computing services, etc.) problems, the model service composition is special. It needs to be validated and evaluated in the whole lifecycle of the model to ensure the credibility of the entire composition model. Therefore, in future research, except the

indicator of model maturity, a study on the other QoS characteristics of M&S is required. From the algorithm's perspective, parameter adjustment and code refactoring are required to improve the operational efficiency of the proposed CA-AO-NSGA-II. Moreover, more variable objective functions of model maturity need to be explored further in order to find appropriate ways to evaluate it.

Acknowledgements

This work is partly supported by National Key R&D Program of China (2018YFB1701600) and by the National Natural Science Foundation of China (NSFC) under Grant Nos.61873014 and 61973243.

8. REFERENCES

[1] Mittal, Saurabh, J. L. Risco-Martin, and B. P. Zeigler, DEVS/SOA: A cross-platform framework for net-centric modeling and simulation in DEVS unified process, Simulation: Transactions of The Society for Modeling and Simulation International, 85(7) (2009) 419-450.

[2] Song, X, Zhang, L, He, DJ, Ren, ZY, A DEVS Based Modelling and Methodology-COSIM, Applied Mathematics & Information Sciences, 6(2) (2012) 417-423.

[3] Song, Xiao, F. Li, and L. Zhang, A Survey and Preliminary Research on Service Federation Based Modeling and Simulation, 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC) IEEE, 2015.

[4] Byrne, James, C. Heavey, and P. J. Byrne, A review of Web-based simulation and supporting tools, Simulation Modelling Practice & Theory, 18(3) (2010) 253-276.

[5] Li Bohu et al, Networked Modeling & Simulation Platform Based on Concept of Cloud Computing Cloud Simulation Platform, Journal of System Simulation (in Chinese), 21(17) (2009) 5292-5299.

[6] Liu, Xiaocheng, et al, Cloud-Based Simulation: The State-of-the-Art Computer Simulation Paradigm, Acm/ieee/scs Workshop on Principles of Advanced & Distributed Simulation ACM, Zhangjiajie, China, 2012.

[7] Cayirci, E, Modeling and simulation as a cloud service: a survey, In Proc of. the 2013 Winter Simulation Conference, WSC'13, San Diego, CA, USA, 2013, pp. 389-400.

[8] Wei-Tek Tsai et al, SimSaaS: simulation software-as-a-service, Int. Conf. Spring Simulation Multi-conference DBLP, 2011, pp. 77-86.

[9] Zhang, Lin, Ying, Liu, et al, Model maturity towards modeling and simulation: Concepts, index system framework and evaluation method, International Journal of Modeling Simulation and scientific Computing, 11(3) (2020) 2040001.1-21.

[10] Marvasti, A. K., et al, Optimal Operation of Active Distribution Grids: A System of Systems Framework, IEEE Transactions on Smart Grid, 5(3) (2014) 1228-1237.

[11] Y. Wu, X. Song, G. Gong, Real-time load balancing scheduling algorithm for periodic simulation models, Simulation Modelling Practice and Theory, 52(1) (2015) 123-134.

[12] Friedenthal S, Alan M, 2011. A Practical Guide to SysML: The Systems Modeling Language. In: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. https://doi.org/10.1115/DETC2004-57751.

[13] Brutzman D, Tolk A, Jsb composability and web services interoperability via extensible modeling simulation framework (xmsf), model driven architecture (mda), component repositories, and web-based visualization. Technical Report. U.S. Air Force, Joint Synthetic Battlespace Analysis of Technical Approaches (ATA) Studies Prototyping, USA, 2003.

[14] Wittman R, Harrison C, Onesaf: A product line approach to simulation development. Technical Report. MITRE CORP ORLANDO FL, USA, 2001.

[15] Petty M, Weisel E, A formal basis for a theory of semantic composability. Proceedings of the Spring 2003 Simulation Interoperability Workshop, Kissimmee, FL, 2014, pp.416-423.

[16] Xiaoyu Kang, et al, CPN-Based Composition in Modeling Command and Control of Surface Air Defense, Communications and Information Processing, Springer Berlin Heidelberg, 2012.

[17] Alpdemir, M Nedim, SiMA: a discrete event system specification-based modelling and simulation framework to support model composability, Journal of Defense Modeling & Simulation, 9(2) (2012) 147-160.

[18] Mittal, Saurabh, J. L. Risco-Martin, and B. P. Zeigler, DEVS/SOA: A cross-platform framework for net-centric modeling and simulation in DEVS unified process, SIMULATION: Transactions of The Society for Modeling and Simulation International, 85(7) (2009) 419-450.

[19] Petty, Mikel D., et al, Software frameworks for model composition, Modelling and Simulation in Engineering, 2014(2015):4.

[20] Taylor, S. J., T. Kiss, A. Anagnostou, et al, The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations, Future Generation Computer Systems, 88(1) (2018) 524-539.

[21] Wang, Sixuan, and Gabriel Wainer, Modeling and simulation as a service architecture for deploying resources in the Cloud, International Journal of Modeling, Simulation, and Scientific Computing, 7(1) (2016) 1-38.

[22] Balalaie, Armin, A. Heydarnoori, and P. Jamshidi, Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, IEEE Software, 33(3) (2016) 42-52.

[23] Wainer, Gabriel, and S. Wang, A Mashup Architecture with Modeling and Simulation as a Service, Journal of Computational Science, 21(2015) (2015) 113-131.

[24] Wainer, Gabriel, and S. Wang, MAMS: Mashup architecture with modeling and simulation as a service, Journal of Computational Science, 21(2017) (2017) 113-131.

[25] Alrifai, M., et al, Ahybrid approach for efficient Web service composition with end-to-end QoS constraints, ACM Transactions on the Web, 6(2) (2012) 1–31.

[26] Chhun, Sophea, Moalla, Néjib, and Y. Ouzrout, QoS ontology for service selection and reuse, Journal of Intelligent Manufacturing, 27(1) (2016) 187-199.

[27] Liang H, Wen X, Liu Y, 2021. Logistics-involved qos-aware service composition in cloud manufacturing with deep reinforcement learning. ROBOT CIM-INT MANUF, 67(2021) 1-15.

[28] Wang, Fei, Y. Laili, and L. Zhang, A many-objective memetic algorithm for correlation-aware service composition in cloud manufacturing, International Journal of Production Research, (2020) 1-19.

[29] Deng, S. G., et al, Service Selection for Composition with QoS Correlations, IEEE Transactions on Services Computing, 9(2) (2016) 291-303.

[30] Luo, Yihang, Yushun Fan, and Haoyi Wang, Business Correlation-Aware Modelling and Services Selection in Business Service Ecosystem, International Journal of Computer Integrated Manufacturing,26(8) (2013) 772–785.

[31] Hannay, Jo Erskine, and T. V. D. Berg, The NATO MSG-136 Reference Architecture for M&S as a Service, Nato Modelling & Simulation Group Symp on M&S Technologies & Standards for Enabling Alliance Interoperability & Pervasive M&s Applications, 2017.

[32] Shahin, Mojtaba, M. Ali Babar, and Muhammad Aufeef Chauhan, Architectural Design Space for Modelling and Simulation as a Service: A Review, Journal of Systems and Software (2020): 110752.

[33] Mahmood, Imran, et al, An Integrated Modeling, Simulation and Analysis Framework for Engineering Complex Systems, IEEE Access, 99(2019) 1-1.

[34] Eek, Magnus, et al, A Concept for Credibility Assessment of Aircraft System Simulators, Journal of Aerospace Computing Information & Communication, 54(6) (2016) 1-15.

[35] Laili, Yuanjun, Lin Zhang, and Yongliang Luo, Pattern-based validation metric for simulation models, Science China Information Sciences, 63(5) (2020) 159203:1-159203:3.

[36] Liu Ying, Zhang Lin, Yang Yuan, et al. A novel cloud-based framework for the elderly healthcare services using digital twin, IEEE Access, 7(2019) 49088-49101.

[37] Chen Y, Tsai W T. Service-oriented computing and web software integration: from principles to development (5th Edition), Kendall/Hunt Publishing Co., 2015.

[38] Gong, Maoguo, et al, Multiobjective immune algorithm with nondominated neighbor-based selection, Evolutionary Computation, 16(2) (2014) 225-255.