International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET23103171

An Effective Multi-Agent Ant Colony Optimization Algorithm for QoS-Aware Cloud Service Composition

C. Dina [1], A. Suriya[1], P. Madhavan [1] A. Alfred Iraianban [1], Alexander. A [2]
[1]Student, [2]Professor
Department of ECE, Rajiv Gandhi College of Engineering College, Pondicherry, India

ARTICLEINFO

Article History:

Accepted: 10 June 2023
Published: 24 June 2023

Publication Issue
Volume 10, Issue 3
May-June-2023

Page Number
576-602

ABSTRACT

In order to improve the efficiency of wireless network virtualization resource processing, this paper combines the dynamic resource allocation algorithm to construct a wireless network virtualization resource sharing model. This paper proposes a task-oriented resource management model and uses the task-oriented TRS model to describe resources and service processes, reducing the complexity of formulating resource allocation strategies. Moreover, this paper comprehensively considers factors such as centralized coordination control cost and limited domain topology visibility to improve the dynamic resource allocation algorithm. Through the abstraction and unified representation of network resources, resource sharing, and efficient reuse, the coexistence and integration of heterogeneous wireless networks can be realized. Wireless network virtualization can decouple complex and diverse network management and control functions from hardware and extract them to the upper layer for unified coordination and management, thereby reducing network management costs and improving network management and control efficiency

Keywords : AOA, DV-Hop, Global Positioning System, IoT, WSN

I. INTRODUCTION

The composition or the aggregation of services is a process that involves building new services called “composite service”, by assembling existing services, offered by multiple providers, in a workflow process. This process specifies which services are to be invoked in what order and under what preconditions. In composition process a “service” can be “atomic service” or “composite service”. Composition can be either static or dynamic.

A static composition uses atomic services in an unchangeable way depending on the context of the customer. However, there are two main approaches for static composition (orchestration and choreography).

Orchestration: A central coordinator composes a business process of services and is responsible for invoking them and forms a workflow. Common industry standard protocols for service orchestration are:

- XLANG (XML Busines Process Langage) of Microsoft,
- BPML (Business Process Modeling Langage) of BPMI,
- BPEL4WS (Business Process Execution Langage for Web Services), result of the grouping of IBM, Microsoft and BEA, also called BPEL or XSBPEL.

Choreography: Equal parties take part in business collaboration and communicate in a peer-to-peer model. There is no central coordinator. Instead there is a conversation definition that determines the interactions between the participants. Web Services Choreography Description Language (WSCDL) is the corresponding protocol standard which exists in theory but has not been adopted widely in the industry. The common protocols found in the literature for this type of static composition are

- WSFL (Web Service Flow Langage) of IBM,
- WSCL(Web Service Conversation Langage) of HewlettPackard
- WSCI (Web Service Choregraphy Interface) of SUN.

However, this type of composition creates inflexible applications, sometimes inappropriate with customer requirements. Dynamic composition it rather sought in applications with online service customers, which its transactions should be posted in real-time. The composition of services cannot be predefined in advance and will be done at run time also. Most of works in this area, have generally formulated this issue as a problem of discovery of the semantic connection between services. Wherein race is to discover a semantic similarity between the output parameters of first service and the input parameters of next ones and most of them are limited to the functional composition aspects.

Composition structures in a workflow

Independently of composition model (static or dynamic). The composition process is in charge of building, in a workflow, a new composite service, by using atomic services, offered by multiple providers, and should be connected together by different composition structures. Figure shows a composite service example, which demonstrate a brief scenario of a composition process, which the service is followed by and in XOR split composition structure (conditional), with a probability of and respectively, service is followed by either or in AND split structure (parallel), and service is followed by in sequential structure, and may be executed for at most times which represent a loop structure.

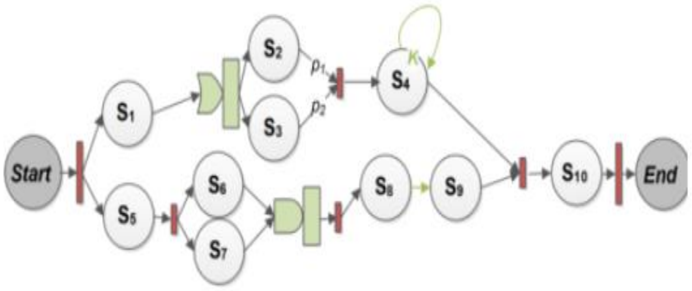


Fig Composite service example

Therefore, we represent hereafter the most composition structures widely considered in literature: Sequential, AND split (fork), XOR split (conditional), Loop, AND join (Merge) and XOR join (Trigger) as shown in Figure

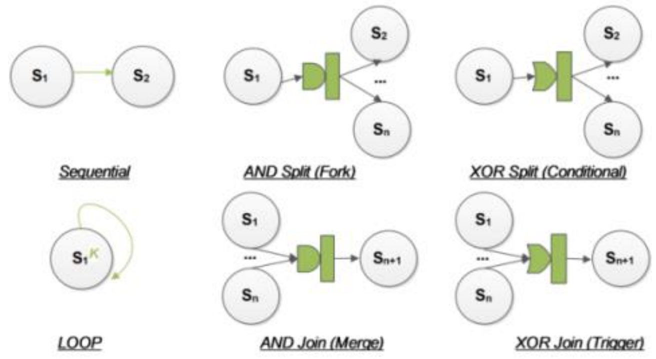


Fig Composition structures in a workflow process

QoS requirement for cloud service

There are several QoS properties defined in standards ISO8402 and ITU E.800, which refer to the nonfunctional properties of services, such as price, response time, availability, reputation, security and so on. They are used to state the quality of services and marked by service providers. In this work four QoS properties are considered. Response Time, Cost, Reliability and Availability, are defined as follows:

Response Time (T): is the time interval from when the service is requested and delivered to the user. It includes the total processing time of the service.

Cost (C): is the cost required to be paid by the customer for the execution of the service. It is considered as an important parameter because certain cloud services cannot be accessed without paying for it and also these Services are costly. The cost of a service is divided in two parts: cost of transmission of request which is omitted in practice, and cost of services.

Reliability (R): is the measurement of the services that correctly serve the users requests.

Availability (A): is the probability that the service is accessible.

II. LITERATURE SURVEY

Title: An Adapted Ant-Inspired Algorithm for Enhancing Web Service Composition

Author: Fadl Dahan, Khalil El Hindi, Ahmed Ghoneim

Introduction

In this work, the authors propose an ant-inspired algorithm for such problem. They named it Flying Ant Colony Optimization (FACO). Flying ants inject pheromone not only on the nodes on their paths but also on neighboring nodes increasing their chances of being explored in future iterations. The amount of pheromone deposited on these neighboring nodes is inversely proportional to the distance between them and the nodes on the path. The authors believe that by depositing pheromone on neighboring nodes, FACO may consider a more diverse population of solutions, which may avoid stagnation. The empirical experiments show that FACO outperform Ant Colony Optimization (ACO) for the WSC problem, in terms of the quality of solutions but it requires slightly more execution time.

Method: FLyING ANT CoLoNy oPTIMIZATIoN (FACo)

The aim of our algorithm is to maintain a good balance between exploration and exploitation. The basic idea can be explained by imaging an imaginary flying ant that deposits its pheromone while flying by injecting it on its path like normal ant. However, since the injection takes place from a distance while the ant is flying, several nodes may get some of the injected pheromone. In other words, not only the nodes on the path would get pheromone, but also their neighboring nodes. These nodes would get an amount of pheromone that is inversely proportional to their distance from the nodes on the path. This gives these neighboring nodes a better chance to be explored in future iterations. Assuming the middle node in Figure is on the best path (local solution) found at a given iteration and has many neighbors. First, our flying ant deposits pheromone on this node, and then distributes the same amount of pheromone between its k nearest neighbor nodes based on the distance between them and the middle node.

FACo for QoS-Aware WSC Problem

To find a solution for WSC using FACO we must address two issues: first, we need to find the best nearest neighbors based on the QoS attributes. Second, we need to inject the right amount of pheromone on the neighboring nodes.

The FACO algorithm has many steps some of them are similar to the original ACO. However, FACO differs from ACO because it needs to search for the neighboring nodes and to inject pheromone on them.

Initially, each ant is randomly placed in a WS in the first task and moves based on heuristic value and past experience. The local update for pheromone is accomplished for each constructed solutions by using Equation. At the end of each iteration, best ant performs the global pheromone update using Equation.

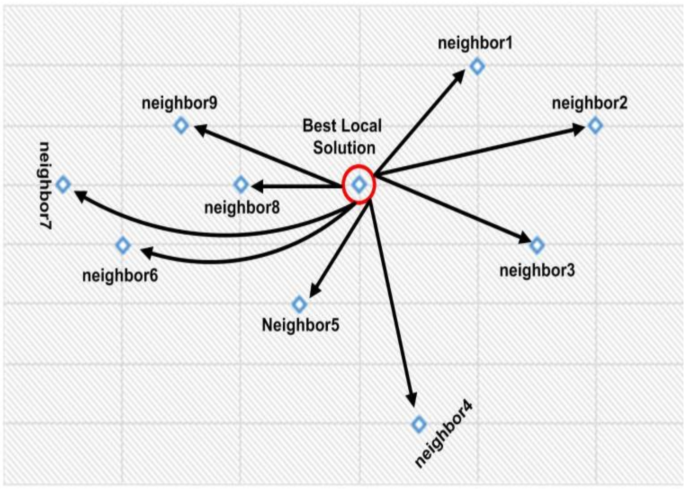


Figure. Flying process to get nearest neighbors

To find the nearest neighbors for QoS-aware composition problem, we use the Euclidean distance to find the nearest WS's to the best solution found based on the QoS values as in Equation

$$d^{(x,y)} = \sqrt{\left(C^{x}-C^{y}\right)^{2}+\left(RT^{x}-RT^{y}\right)^{2}+\left(A^{x}-A^{y}\right)^{2}+\left(R^{x}-R^{y}\right)^{2}}$$

C, RT, A, and R: Cost, response time, availability, and reliability respectively;

x: The best WS getting by the best ant in task ti;

y: Belongs to the set of services in the same task of x, and x≠ y.

The amount of global update is distributed between the k neighbors inversely proportional to their distance from the best WS for each task. Equation is designed to achieve that:

$$\tau_{il}\left(t+1\right)=\tau_{il}\left(t\right)+\left\{\tau_{ij}\left(t+1\right)\Big/\left(1+\sqrt{x'\left(\eta_{jl}\right)}\right)\right\}$$

Where

τij (t+1): The pheromone trails from Equation;

x′(ηji): The normalization for distance between WS j and its neighbor l in the same task and **l ∈ k**

To combine the advantages of normal (waking) ants and flying ants, we assume that only a certain proportion of ants are flying ants and the rest are normal ants. The number of flying ants is determined by the parameter FR (flying ratio)

It is worth noting that the proposed improvement can be used with any ant-inspired algorithm. In addition, it also can be used for local pheromone update, but that would increase the algorithm's complexity.

Maximization Problem

We convert the multi-objective QoS-aware problem into a single objective maximization problem, as follows:

$$argmax_{k\in Ants}(\mathrm{F})=\frac{\left(\mathrm{W}_{a}\prod_{i=1}^{n}A_{ix}^{k}+\mathrm{W}_{r}\prod_{i=1}^{n}R_{ix}^{k}\right)}{\left(\mathrm{W}_{c}\sum_{i=1}^{n}C_{ix}^{k}+\mathrm{W}_{rt}\sum_{i=1}^{n}RT_{ix}^{k}\right)}$$

Minimization Problem

This section presents the problem as a minimization problem. Equation presents the objective function we attempt to minimize. This new formula gives us a new problem specification, which allows us to test the effectiveness and the robustness of the new algorithm

$$argmin_{k\in Ants}(\mathrm{F})=\frac{1}{\sqrt{W_{C}\sum_{i=1}^{n}\left(\frac{1}{C_{ix}^{k}}\right)^{2}+W_{rt}\sum_{i=1}^{n}\left(\frac{1}{RT_{ix}^{k}}\right)^{2}+W_{a}\prod_{i=1}^{n}A_{ix}^{k\,2}+W_{r}\prod_{i=1}^{n}R_{ix}^{k\,2}}}$$

where:

n: The tasks number;

k: Refers to the ant number 1 ≤ k ≤ Ants where Ants is the swarm size;

$$\eta_{ij}=\frac{1}{\sqrt{\frac{1}{C_{ij}^{2}}+\frac{1}{\mathrm{RT}_{ij}^{2}}+\mathrm{A}_{ij}^{2}+R_{ij}^{2}}}$$

$$\Delta\tau_{ij}\left(t\right)=\begin{cases}1\big/f_{best} & \text{if } WS_{ij} \text{ is tbelongs to best path}\\ 0 & \text{otherwise}\end{cases}$$

where:

i:Referstotheithtask;

j:Thejthe WSintotaski;

fbest:ThebestQoSvaluegettingbythebestants.

Title: Enhancement of Ant Colony Optimization forQoS-Aware Web Service Selection

Author: Hashem AlAyed, Fadl Dahan, Taha Alfakih, Hassan Mathkour, Mohammed Arafah

Introduction

In this work, propose an enhancement to the ant colony optimization (ACO) algorithm based on a swap concept for the QoS-aware WSS problem. The aim of the enhancement to the ACO is to avoid the trap of local optima and reduce the search duration. Believe that the integration of many potent solutions will help the ACO algorithm yield a better solution and avoid stagnation.

Method: Proposed Algorithm

In this work, introduce two different enhancements to the ACO algorithm focusing on performance and on overcoming the stagnation problem. First, we introduce a swapping process that lets ACO remember two solutions temporarily and then, in a final step, return the best one only.

This process aims to swap a specific number of web services in the two best solutions found so far to generate two new solutions. The selection of the web services to be swapped is performed randomly. Then, the fitness values of these new solutions are calculated and compared to the best solution remembered by the algorithm. The incorporation of the two best solutions will help the ACO algorithm to get a better solution and avoid the stagnation problem as each one of these solutions can contribute parts that can produce excellent solutions if they are merged

As a second enhancement, we adopted a multi-pheromone version of the ACO algorithm where each QoS constraint (cost, response time, availability, and reliability) is tied to a pheromone value of its own. This improves the chances for ants to consider more options in the solution space (i.e., increases the chance for exploration)

In standard ACO, the pheromone is represented by a single value. In this work, we represent pheromones using multiple values, according to the number of QoS attributes

$$\left[\tau^{C}_{(i,x)(i+1,x')}, \tau^{RT}_{(i,x)(i+1,x')}, \tau^{A}_{(i,x)(i+1,x')}, \tau^{R}_{(i,x)(i+1,x')}\right]$$

where $\tau^{C}_{(i,x)(i+1,x')}$ at task i to WS x' at task i+1 for the cost (C) feature, and RTis for response time, A is for availability and R for reliability. The idea here is to consider the importance of each QoSfeature and to become an effective guide while searching for a better solution.

The transition process is the process of selecting the next node to move on while building the tour. For basic ACO this process is shown in equation. In the proposed algorithm, this equation changes based on the new pheromone representation for ant k at time t, as follows:

$$P^{k}_{(i,x)(i+1,x')}(t) = \frac{\left[\tau_{(i,x)(i+1,x')}\right]^{\alpha}\left[\eta_{(i,x)(i+1,x')}\right]^{\beta}}{\sum_{l\in m}\left[\tau_{(i,x)(i+1,l)}\right]^{\alpha}\left[\eta_{(i,x)(i+1,l)}\right]^{\beta}}$$

Where

$$\tau_{(i,x)(i+1,x')} = \tau^{A}_{(i,x)(i+1,x')} + \tau^{R}_{(i,x)(i+1,x')} \mp \tau^{C}_{(i,x)(i+1,x')} + \tau^{RT}_{(i,x)(i+1,x')}$$

$$\eta_{(i,x)(i+1,x')} = (\eta^{A}_{(i,x)(i+1,x')} + \eta^{R}_{(i,x)(i+1,x')}) - (\eta^{C}_{(i,x)(i+1,x')} + \eta^{RT}_{(i,x)(i+1,x')})$$

The new formula helps ants to consider the value of the QoS features individually. This technique allows ants to explore the search space efficiently compared to the single-pheromone aggregation of these features. When an ant k moves from each task to the next it updates the pheromone locally on its path. In the proposed algorithm, the updating changes to consider the new pheromone distribution based on the QoS features, the local pheromone update for the cost feature.

$$\tau^{C}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{C}_{(i,x)(i+1,x')}(t) + \rho\tau^{C}_{0}$$

The local pheromone update for the response time feature is shown

$$\tau^{RT}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{RT}_{(i,x)(i+1,x')}(t) + \rho\tau^{RT}_{0}$$

the local pheromone update for the availability feature.

$$\tau^{A}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{A}_{(i,x)(i+1,x')}(t) + \rho\tau^{A}_{0}$$

the local pheromone update for the reliability feature.

$$\tau^{R}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{R}_{(i,x)(i+1,x')}(t) + \rho\tau^{R}_{0}$$

At the end of each iteration, all ants complete the construction of the possible solutions and then calculate the fitness of their solutions. Each possible solution contains n concrete web services adhering to the abstract definition of the workflow. The calculation formula for the fitness of the ants' solution is:

$$F = \left(\prod_{i=1}^{n} A^{k}_{(i,x)(i+1,x')} + \prod_{i=1}^{n} R^{k}_{(i,x)(i+1,x')} - \sum_{i=1}^{n} C^{k}_{(i,x)(i+1,x')} - \sum_{i=1}^{n} RT^{k}_{(i,x)(i+1,x')} \right)$$

Where n is the number of tasks. K refers to the ants' number with 1≤k≤S and S is the swarm size.

The ACO memorizes the best solution at each iteration and allows the best ant to update the amount of pheromone on its path. Therefore, the global pheromone update formula also changes to consider QoS features, as does the local pheromone update. the global pheromone update for the cost feature.

$$\tau^{C}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{C}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{C}_{(i,x)(i+1,x')}(t)$$

where:

$$\Delta\tau^{(i,x)(i+1,x')}_{C} = \begin{cases} \dfrac{1}{\sum_{i=1}^{n} C^{(i,x)(i+1,x')}_{best}} & if\ WS_x \text{ and } WS_{x'} \in best\ path \\ 0 & otherwise \end{cases}$$

Equation shows the global pheromone update for the response time feature

$$\tau^{RT}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{RT}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{RT}_{(i,x)(i+1,x')}(t)$$

where:

$$\Delta\tau^{A}_{(i,x)(i+1,x')} = \begin{cases} \prod_{i=1}^{n} A^{best}_{(i,x)(i+1,x')} & if\ WS_x \text{ and } WS_{x'} \in best\ path \\ 0 & otherwise \end{cases}$$

Equation shows the global pheromone update for the reliability feature.

$$\tau^{R}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{R}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{R}_{(i,x)(i+1,x')}(t)$$

where:

$$\Delta\tau^{R}_{(i,x)(i+1,x')} = \begin{cases} \prod_{i=1}^{n} R^{best}_{(i,x)(i+1,x')} & if\ WS_x \text{ and } WS_{x'} \in best\ path \\ 0 & otherwise \end{cases}$$

In the proposed algorithm, postpone the global pheromone update to the end of the swapping process and select the two best ants for the swapping process; this process aims to merge the current two solutions to produce two new solutions. Figure presents the swapping process used in the proposed algorithm. The proposed algorithm remembers the two best solutions for swapping. These two solutions are the two best solutions discovered globally. The number R of web services to be swapped is generated randomly within [1,n] where n is the number of tasks. The selection of web services to be swapped is performed randomly. The swapping involves the web services in the first solution and the web services in the second solution.

Title: Two-Step Artificial Bee Colony Algorithm Enhancement for QoS-Aware Web Service Selection Problem

Author: FADL DAHAN, HASSAN MATHKOUR AND MOHAMMED ARAFAH

Introduction

This paper presents an enhanced artificial bee colony (ABC) algorithm for solving the web service selection problem. The proposed algorithm searches the best possible combination of web services to satisfy user requirements. An adapted neighborhood selection and replacement process and a swapping process are used to improve the ABC behavior. Neighboring nodes are employed to enhance ABC performance by encouraging exploration in early iterations, where bees have no knowledge regarding the search space, and by encouraging exploitation in later iterations to exploit bee knowledge of the search space. The swapping process is used to enhance ABC performance by randomly swapping portions among the best two solutions randomly.

Method: The Proposed Two-Step ABC

In this study, we introduce an additional enhancement to the enhanced ABC (EABC) algorithm. This enhancement introduces a two-step search process. In the first step, the proposed algorithm uses the EABC selection and replacement strategies. The search process using employed and onlooker bees were modified to adopt selection and replacement strategies. The selection strategy picks a task from the best solution to replace one web service by another web service based on the distance between the neighboring web services SN from this task. The value of SN is experimentally tested based on to be equal to 5. To do this, the selection strategy selects a random task ti from the generated solutions. One of this task's web services is replaced by a neighboring web service of this task. Euclidean distance is used to calculate the distance to the randomly selected SN neighbors, as shown in equation. Figure presents the selection and replacement strategies in details.

$$d_{xj} = \sqrt{(C_x - C_j)^2 + (RT_x - RT_j)^2 + (T_x - T_j)^2 + (R_x - R_j)^2}$$

$$j \in [1, S_N]$$

where C_x and C_j are the web services' (x and j) cost; RT x and RT j are the web services' (x and j) response time; T_x and T_j are the web services' (x and j) throughput; and R_x and R_j are the web services' (x and j) reliability. SN is the number of web services' neighboring nodes.

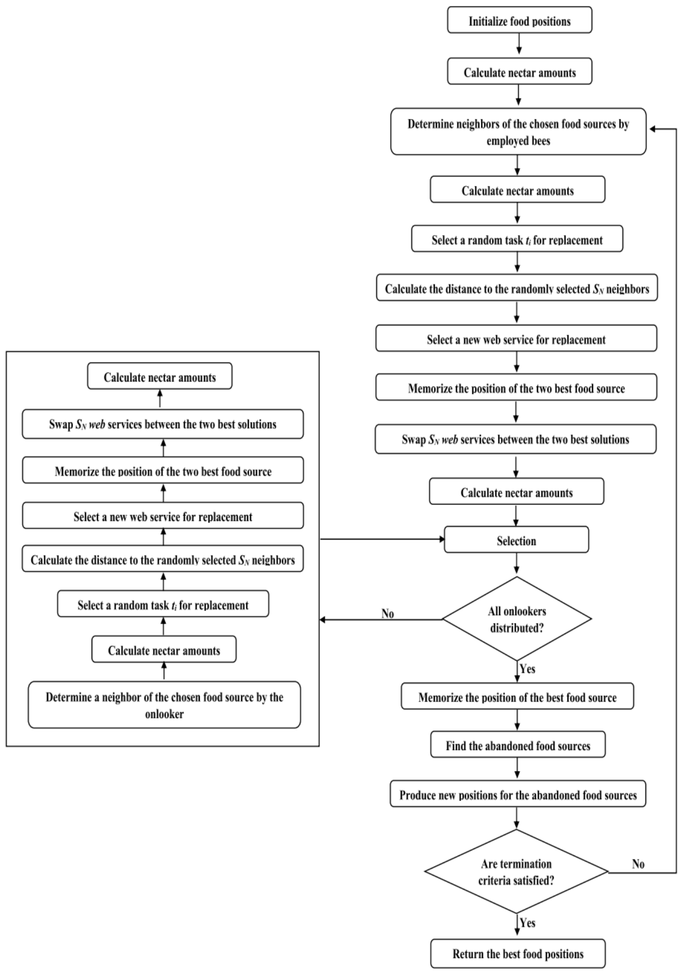

Initialize food positions
Calculate nectar amounts
Determine neighbors of the chosen food sources by employed bees
Calculate nectar amounts
Select a random task t_i for replacement
Calculate the distance to the randomly selected S_N neighbors
Select a new web service for replacement
Memorize the position of the two best food source
Swap S_N web services between the two best solutions
Calculate nectar amounts
Selection
All onlookers distributed?
No
Yes
Memorize the position of the best food source
Find the abandoned food sources
Produce new positions for the abandoned food sources
Are termination criteria satisfied?
No
Yes
Return the best food positions
Calculate nectar amounts
Swap S_N web services between the two best solutions
Memorize the position of the two best food source
Select a new web service for replacement
Calculate the distance to the randomly selected S_N neighbors
Select a random task t_i for replacement
Calculate nectar amounts
Determine a neighbor of the chosen food source by the onlooker

Figure The proposed algorithm flowchart

The distances of selected neighbors and the best web service are sorted. To select a new web service for replacement, use

$$index = \left(\frac{rand}{i}\right) * (S_N)$$

where i is the iteration number, rand is uniform random number and SN is the number of neighboring web services from same task ti.

Equation aids the proposed algorithm to maintain a balance between exploration and exploitation. In the early iterations, the algorithm selects a web service randomly from the farthest neighbors for replacement. In later iterations, it selects web service randomly from the nearest neighbors. This process encourages exploration in early iterations and exploitation in later iterations.

In the second step, introduce a swapping process for the two best solutions. The aim of this process is to swap between the web services of two solutions to generate two new solutions. As illustrated in Figure 6, the proposed algorithm remembers the two best solutions for swapping. These two solutions are the two best solutions discovered globally. The number of web services to be swapped is similar to the number of neighbors in the first process (SN). The selection of web services to be swapped are performed randomly. The result of this process is two new solutions. Then calculate the fitness values of these solutions and compare them to the best solution remembered by the algorithm. The algorithm retains only the best solution for further iteration.

Solutions' fitness values are computed using preference-based method. In preference-based method, a multi-objective problem is converted into a single-objective problem using an aggregation function. The aggregation function used in the proposed method is shown below

$$F = (\prod_{i=1}^{n} T_{ib} + \prod_{i=1}^{n} R_{ib} - \sum_{i=1}^{n} C_{ib} - \sum_{i=1}^{n} RT_{ib})$$

where C refers to cost, RT refers to response time, T refers to throughput, and R refers to reliability. n refers to the number of tasks; and b refers to the employed or onlooker bees' selection for a web service to task i in each iteration.

The two steps above are utilized by both employed and onlooker bees. However, employed bees should recruit the onlooker bees to follow them based on the fitness values of their solutions using equation. Any employed bees that find no improvements for their solutions are reinitialized as scout bees. The flowchart of the proposed algorithm is shown in Figure. Figure presents the proposed algorithm in details.

Title: QoS-aware cloud service composition using eagle strategy

Author: Siva Kumar Gavvala, Chandrashekar Jatot, G.R. Gangadharan and Rajkumar Buyya

Introduction

In this work, propose an algorithm that uses Eagle Strategy allowing us to authorize both exploration and exploitation in an effective way to balance the foraging process.

Methods

Eagle Strategy with Whale Optimization Algorithm (ESWOA)

This section describes our proposed Eagle Strategy with Whale Optimization Algorithm (ESWOA).

Eagle strategy

Eagle strategy technique was developed by Yang and Deb that does optimization in two phases, preserve the balance between exploration and exploitation. In this strategy, the exploration is done similar to how an eagle searches for its prey initially. Once the prey is found the eagle changes its behavior in chasing the prey to intensive attacking. This has been imitated by this strategy in the exploitation phase, by integrating an optimization technique that does a rigorous local search such as downhill simplex or Nelder–Mead method. Evidently, we could use various efficient meta-heuristic algorithms like Particle Swarm Organization, Firefly Algorithm Differential Evolution or Artificial Bee Colony to do a strenuous local search. In Levy walk and Firefly algorithm have been coupled to draft a Eagle Strategy technique. The pseudocode for Eagle strategy is presented in Algorithm

The parameter Pe enables us to authorize in an iterative manner between the exploration and exploitation. To start with, initial solutions are mounted from a large search space as these solutions often constitute high diversity. These instances undergo an evolution by an intensive metaheuristic algorithm that leads to a converged state, the state in which solutions have low diversity. Subsequently, a new set of solutions are acquired again from the larger search space that again comprises of high diversity, for another round of intensive iteration stage. In a similar manner, exploration and exploitation have been utilized to preserve the superior degree of diversity in the entire population

Whale optimization algorithm (WOA)

WOA is one of the latest metaheuristic algorithms introduced by Mirjalili et al. in 2016. WOA algorithm mimics the behavior of the Humpback whales. Usually, these whales wander in groups and often behave intelligently to catch their prey, which is called the bubble-net attacking method as shown in Figure, a strategy that hunts small fishes by trapping them in self-created distinctive bubbles along a circle or 9-shaped path. This foraging has been mimicked in the WOA algorithm.

To replicate this behavior in WOA, there is 50% probability that the whale follows any of the two paths (circle or 9-shaped paths) mentioned to update the position of the whale during optimization. The mathematical representation of the WOA strategy is given as follows:

Encircling prey mechanism

In WOA, make an assumption that the current best solution is the target prey or near optimal solution, while other whales try to update their positions in the direction of this current best solution. This is represented as follows:

$$\vec{D} = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)|$$

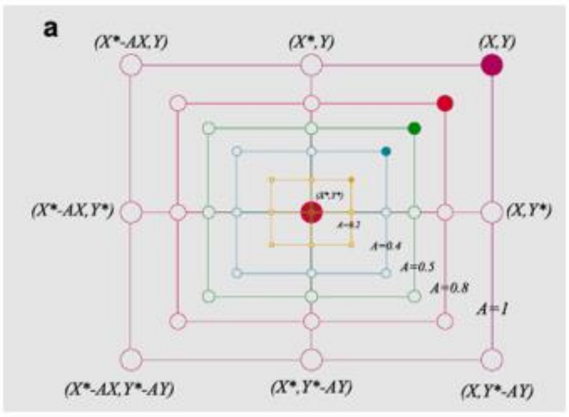


Fig. Shrinking encircle mechanism

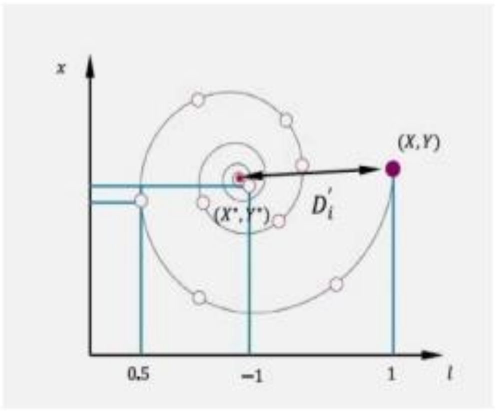


Fig. Spiral updating position

$$\vec{X}(t+1) = |\vec{X}^*(t) - \vec{A} \cdot \vec{D}|$$

where X⃗∗ denotes the position of best whale found till now, X⃗ is the position of the whale at current iteration and t represents the current iteration.

$$\vec{A} = 2.\vec{a} \cdot \vec{r} - \vec{a}$$

$$\vec{C} = 2 \cdot \vec{r}$$

a⃗ will be decreased from 2 to 0 linearly in every iteration and ⃗r is the random number in the range [0, 1].

Bubble net attacking method (exploitation)

Shrinking Encircling: We decrease the value of a linearly that in turn decreases A⃗ value, as it ensures that the newly updated whale will fall in the range of [-a, a]. Now, the new possible positions of the individual whale that will be determined in between the current whale position and best whale position is shown in Figure in a 2-Dimensional space. Where (X, Y) denotes the position of current selected whale and (X∗, Y ∗) represents the best whale position

Spiral Updating Position: The current whale follows the best whale by moving in an helix shaped path as shown. This figure also shows the new possible positions of whale that is going to be updated. A mathematical equation represented for this helix shaped behavior, which is used to update the position between the current whale and the target whale

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t)$$

where l represents a random number within the range [-1, 1], b ensures the logarithmic shape and D⃗′ is the distance the between current whale and the prey.

Search for prey (exploration)

Unlike in encircling prey mechanism that updates the current whale position based on the best whale, here updating is done based on the random whale which is decided by the vector A⃗. Exploration follows

$$\vec{D} = |\vec{C} \cdot \vec{X}_{Rand} - \vec{X}|$$

$$\vec{X}(t+1) = |\vec{X}_{Rand} - \vec{A} \cdot \vec{D}|$$

where X⃗Rand is the random whale selected. This whale is used as a reference to update the current selected whale. The pseudocode of Whale Optimization

Before proceeding to proposed method, we present the correspondence (Mapping) between Whale Foraging and service composition for better understanding

Eagle Strategy with Whale Optimization Algorithm (ESWOA)

In our proposed technique, apply the exploitation process of the WOA algorithm to perform only the local search that ensures the intensification part of the Eagle strategy. Apply a new approach for the exploration purpose that performs the diversification part of our Eagle Strategy. Explain the phases starting from encoding to obtaining optimal solution as follows.

Encoding

In traditional WOA algorithm, the whale position represents the feasible solution to an optimization problem that is denoted as a m-dimensional vector. The whale that has the highest fitness represents as a Leader Whale, and the fitness is denoted by the Leader score (X∗), and its position is represented as the Leader Position (LP). We need to maintain the ranges of each dimension for our candidate solution depending on the requirement, with respect to our service composition problem. Therefore, the encoding of whale position should consider the procedure for producing the candidate solutions according to our necessity.

The main goal of QoS-aware cloud service composition is to select a service from a pool of candidate services wsi,1, wsi,2, wsi,3, . . . , wsi,nm of each abstract service Si, obeying the QoS constraints, which finally result in the maximizing the quality of composite service. We adopted an integer encoding scheme similar that maps an integer to the concrete service. A whale position x_d is represented as a m-dimension vector x_d = {x1 d, x2 d, x3 d, . . . , xm d }. In this array of numbers, each element $x_{i\,j}$ represents the value of the jth concrete service from the ith abstract service. This value is bound to be in the range of [lb, ub], where lb is 1 and ub is number of concrete services in the pool Si.

Initialization

In WOA, we generate SN number of whale positions randomly, represented as {x1, x2, x3, . . . , xSN}

$$x_d^i = lb + \lfloor rand(0, 1) * (ub - lb) \rfloor$$

where SN represents the number of candidate solutions in our search space (population). Now, for every candidate solution, calculate their fitness values csQoS using the Eq. (Line 1 and 2 in Algorithm). We will get SN fitness values, out of which the highest fitness value is treated as Leader score (X∗) (Line 3 in Algorithm) and the corresponding whale position is treated as Leader position

Let us assume that we have m = 5 and ni = 5 where 1 ≤ i ≤ 5 and SN is 6. The sample population is generated randomly satisfying the said constraints and their respective fitness values (csQoS) are calculated by Eq. For example [3, 2, 1, 2, 2] means the selection of ws1,3, ws2,2, ws3,1, ws4,2, ws5,2. From these, the highest csQoS is 1.5791 and is stored in X∗ and its corresponding position is stored as the Leader position.

Iteration process

After initialization, the candidate solutions will undergo exploration and exploitation processes for MAX_Iter times in search of better solutions than X∗. Let MAX_Iter be the maximum number of iterations that the algorithm runs.

Exploration phase in eagle strategy

In this phase of our proposed algorithm, each dimension of the whale position is changed according to the probability prob

$$prob = 0.3(1 - \frac{iter}{MAX\_Iter})$$

where iter is the current iteration number and MAX_Iter is the maximum number of iterations specified initially.

Now, for a currently selected whale, we generate a random number within the range of [1, dim], to determine which dimension should be randomly changed. Another random number q is extracted within the interval [0, 1] and is compared with the probability prob. If q<prob, then a modification is done to the dimension

$$X_j = X_{jmin} + rand.(X_{jmax} - X_{jmin})$$

where rand is a random number generated within the range [0,1]. X_j is the selected dimension that is to be altered, X_{jmax} is the maximum of dimensions from the currently selected whale, X_{jmin} is the minimum of dimensions.

Title: Cloud service composition using an inverted antcolony optimisation algorithm

Author: Saied Asghari and Nima Jafari Navimipour

Introduction

This paper proposes the inverted ant colony optimization (IACO) algorithm, a variation of the basic ant colony optimization (ACO) algorithm, to solve this problem. This method inverts its logic by converting the effect of pheromone on the selected path by ants in order to improve load balancing among cloud servers. In this method, ants begin to traverse the graph from the start node and each ant selects the best node for moving, then other ants may not follow the track travelled by the previous ants.

III. Method : Proposed method

Problem definition

Definition 1: (MCE): An MCE is a set of clouds, such that MCE = {C1, C2, …, Cn}, in which each of the MCE components is an independent cloud server or a cloud. The multi-cloud scheme is the concomitant use of two or more cloud services to minimise the risk of widespread data loss or interruption due to a localised component failure in a cloud .

Definition 2: (cloud server): A cloud server is a set of service files in the form of C = {sf1, sf2, …,sfn}. In this method, each cloud server is shown as a node in the graph.

Definition 3: (service file): A service provider publishes a set of web services, which corresponds to the service file sf in the form of sf = {w1, w2, …,wn} and each of the sf components is a web service. Each service provider is a service file.

Definition 4: (web service): A web service includes a two-tuple in the form of <1, 0> in which both 1 and 0 are service interfaces that are sets of propositions provided by user's request. In this work, we consider that an interface consists of ontology concepts

A framework of the proposed method is illustrated in Figure. An MCE is made up of several parts, including the interface, cloud combiner, and composition planner. A user sends a request to the cloud combiner through the interface.

All service files are available in the interface which recognises the needed service files to users for response. The cloud combiner tries to select the optimal cloud composition that satisfies user needs. Using an optimization technique, the cloud combiner chooses an appropriate cloud composition from the MCE. Finally, the web service composition sequence is generated by the composition planner and is sent to the user.

Service composition using an IACO algorithm

At first step, for dealing with the problem of a cold start (at the beginning of the graph traversal, there are no pheromones spread in the graph), equation (1) is used. Therefore, we will acquire the minimum number of clouds that satisfy user requirements. In each iteration, ants select the next cloud server among cloud servers having more probability using equation. However, our heuristic approach is different causing the performance of our method to be more efficient in the use of web services which causes cost reduction in some cases. Although the method for selecting a cloud server considers spent cost, in ACO method, the spent cost for selecting a cloud server will not be considered and the heuristic value is different.

In the proposed method, each ant selects its path according to the pheromone, heuristic information, and profit amount on the edges of the path. The pheromone on each edge is updated according to the quality of the paths. In this method, ants begin the graph traversal from the start node and the best node is selected by the first ant to move and the pheromone is updated according to equation which causes IACO to improve load balancing among cloud servers.

Therefore, the track travelled by the previous ant might not be followed by the second ant since the available pheromone in the edge travelled by the previous ant has caused aversion. In other words, ants in this method try to select the new node to travel among nodes that have not already been visited. If the calculated probability for a cloud server is zero, the cloud server will not be selected by any ant. If two nodes exist in the same condition, one of them will be selected randomly. If all of the available edges of each node are traversed to satisfy the requirements, other ants do duplicate and select the traversed path in order:

$$p_{kj}^{i}(t) = \frac{[\tau_{kj}(t)^{\alpha}][\eta_{kj}(t)^{\beta}][\theta_{j}^{i}(t)^{\gamma}]}{\sum_{l=1}^{n} p_{kl}^{i}(t)[\tau_{kl}(t)^{\alpha}][\eta_{kl}(t)^{\beta}][\theta_{j}^{i}(t)^{\gamma}]}$$

where t is the current iteration, $\tau_{kj}(t)$ is the pheromone on each edge e_{kj}, η_{kj} is the heuristic information on edge e_{kj} and finally θ_{ij} (t) is the gain for ith ant selecting the jth cloud.

Parameters $\alpha, \beta$and γ are values that show the amount of impress of the pheromone, heuristic information, and the gain. In our work, according to most of the previous methods their value is one:

$$\tau(t+1) = \tau(t) \times \frac{1}{k}$$

The above-mentioned relationship represents the pheromone update in each iteration. The amount of pheromone (t(t)) is one at first in all edges. The heuristic information on edge e_{kj} is defined as:

$$\eta_{kj} = \frac{1}{f}$$

f in the above formula is the number of web services available in a cloud. Equation (4) (Yu et al., 2015) shows that the gain θij ( ) t for the ith ant selecting the jth node is defined as the number of required service files in the jth node:

$$\theta_{j}^{i}(t)|C_{j} \cap S_{r}(i)|$$

In the following, the considered parameters are going to be defined so that the method be evaluated in comparison with the other algorithms.

Definition 5: (optimal cloud composition): Optimal cloud composition is the minimum number of cloud servers that can be combined together and satisfy user requirements.

Definition 6: (load balancing): Through load balancing the amount of work that a processing node has to do between two or more nodes is divided so that more work gets done in the same amount of time and, in general, all users get served faster

Definition 7: (waiting time): Another parameter called the waiting time is considered to compare algorithms. Each user's request is referred to as a task and it is assumed that each ant performs a user's request. In this regard, the waiting time refers to the time that each task spends to utilize the cloud and is assumed that 1 ms time is needed for the task to collect service files required from a cloud. It is assumed that if a common cloud is needed to respond user requests, this cloud will be provided to the users in order; thus other users will wait for 1, 2, 3 and … ms, respectively. The waiting time for each user is calculated according to equation in which n is the number of clouds and t is time that the task spends to take control cloud. Equation describes total time the users have spent to meet their needs in which T is the number of tasks or users and ui is the obtained waiting time for each user

Waiting time for each user: $\sum_{i=1}^{n} t \times C_i$

Waiting time for all users: $\sum_{i=1}^{T} u_i$

Definition: (cost): equation shows the spent cost for creating service composition sequence where c is the number of available clouds in optimal cloud composition and f is the number of web services available in a cloud. It is assumed that in order to evaluate the condition of each web service by composition planner one dollar is needed:

$$cost = \sum_{i=1}^{c} f$$

The descriptions of the algorithm for the proposed method are included in the following lines. Inputs consist of the number of ants, the number of clouds, received a request by the user of the system and the available data in each cloud

IV. Existing System

Problem Formulation

The providers target the enterprises'/users' requests satisfaction by providing enormous services over the clouds. These requests can be achieved by a service (primitive task) or a set of services (composite task).

The composite task can be defined as a task that can be decomposed into a set of primitive tasks based on its functionality. For this decomposition, the composite task can be orchestrated into workflow where each subtask includes a definition of a primitive task. The overview of CSC framework is represented in Figure

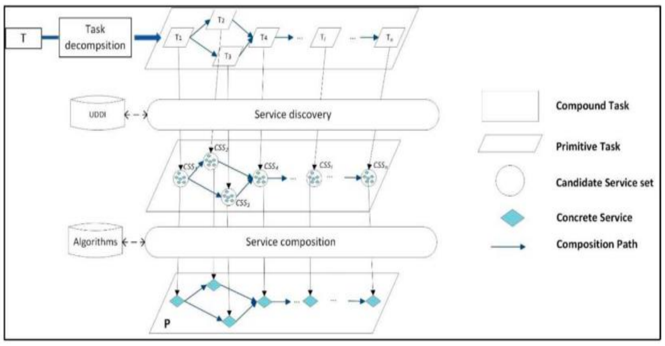


Figure CSC Framework Overview

In the CSC framework, DAG is used to model the workflow. The workflow consists of a set of tasks (n tasks), and for each task, a candidate services list (m services) satisfies the functionality of ith task. The DAG representation for the CSC problem captures the relative information of candidate list among the subtasks and represents this by G = (V, E, P, C, W):

$v = \{T_i | i = 1:n\}$ is the set of tasks and n is the number of tasks in the workflow. The candidate list is represented for m services

$$(CS_i^1, CS_i^2, \ldots, CS_i^m),$$

where $CS_i^j (1 \leq i \leq n, 1 \leq j \leq m)$ is the jth service for ith task.

E is the edges among of candidate list and represents the QoS properties, such as cost and availability. The QoS properties Q is represented for the jth service as follows:

$$(qos_{1,i}^j, qos_{2,i}^j, \ldots, qos_{r,i}^j),$$

where $qos_{r,i}^j (1 \leq r \leq Q)$ is rth QoS attribute of the jth service for ith task.

P is the structural patterns of the workflow, and includes sequential, parallel, conditional, or loop patterns.

C is the enterprises'/users' constraints set over the QoS properties where $C = \{C_1, \ldots . C_Q\}$.

W is the enterprises'/users' preferences for the QoS properties where $W = \{w_1, \ldots . w_Q\}$.

Given the above DAG, the DAG has a set of possible paths that can fulfill the enterprises'/users' requests. Therefore, the CSC can be defined as finding the superior path among these possible paths under the QoS properties. The CSC is multi-objective because the objectives of some of the QoS properties need to be minimized (called negative QoS, e.g., cost, and response time) while others need to be maximized (called positive QoS, e.g., availability, and reliability). Also, it is multi-constraint as it seeks to satisfy the lower and upper bound of the QoS properties. In this work, we regard the CSC problem as multi-objective optimization problem; therefore, this problem can be represented as a mathematical model:

$$\text{Max}F = \sum_{r=1}^{Q} W_r * N\left(qos_r\right)$$

where W_r is the client's weight for the rth QoS property where the sum of the weights for all QoS properties is equal to 1, and $N(qos_r)$ is the normalization of the value of QoS properties to guarantee the maximum optimization of CSC problem. The normalization of negative and positive QoS properties is shown

$$N\left(qos_r\right) = \begin{cases} \dfrac{Q_r - minQ_r}{maxQ_r - minQ_r}, , & if\, maxQ_r \neq minQ_r \\ 1, & otherwise \end{cases}$$

$$N\left(qos_r\right) = \begin{cases} \dfrac{maxQ_r - Q_r}{maxQ_r - minQ_r}, & if\, maxQ_r \neq minQ_r \\ 1, & otherwise \end{cases}$$

where $minQ_r$ is the minimum value of the rth QoS attribute, whereas $maxQ_r$ is the maximum value of the rth QoS attribute.

Four patterns are available for the workflow (i.e., sequential, parallel, conditional, and loop) that depend on the execution of the enterprises'/users' requests. The aggregated value of the QoS properties. These formulas represent the worst-case calculation of the workflow

The CSC seeks to find a superior path among DAG possible paths based on the QoS properties. Therefore, it is a multi-objective optimization problem with NP-hard complexity. The nature-inspired algorithms do not suffer from the computational restriction because they search not for an optimal solution, but for the near-optimal solution

ACO for CSC Problem

Ant Colony Optimization

The ACO is mainly introduced to solve combinatorial optimization problems type where problems are represented as a graph G =(V,E), where V represents the set of vertexes (nodes) and E represents a set of edges. Initially, all the graph edges have the initial amount of pheromone trail T 0, although the Z ants move through the graph nodes to find the first set of solutions. The initial solution transition rule is given in Eq. Then, a set of process is iterated until a stop condition is met. These processes include searching for new solutions, updating pheromone trails, evaluating new solutions, and memorizing the best solution.

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in\mathcal{N}_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \quad if\ j \in J_k(i)$$

where α and β determine the contribution level of pheromone and heuristic coefficients, respectively. T_{ij} is the pheromone trail on edges i and j. r_j is the problem heuristic. $J_k(i)$ is the list of unvisited nodes by ant k

The ACO family includes different algorithms that have different selection/updating strategies for selecting the next node and updating pheromone trails during searching for new solutions. In this work, we focus on ACS to solve the CSC problem.

Ant Colony System (ACS)

This rule is used to construct piece-by-piece solutions from both the ants' previous knowledge (pheromone trail τ_{ij}) and the problem heuristic (η_{ij})

$$P_{ij}^{k}(t) = \begin{cases} \underset{j\in J_k(i)}{\arg\max}\left\{[\tau_{ij}]^{\alpha}\cdot[\eta_{ij}]^{\beta}\right\}, & if\ q \le q_0(exploitation) \\ P_{ij}^{k}, & otherwise(exploration) \end{cases}$$

where the meaning of α, β, and $J_k(i)$ is found in Eq. The q parameter is a random value between 0 and 1, and the q0 parameter is $0 < q0 < l. P_{ij}^k$is obtained

The piece of solution given by Eq. represents a tradeoff between exploitation and exploration. The value of q0 is the balance of this tradeoff, where exploitation is favored for selection if q < q0; otherwise, exploration is favored

In ACS, there are two different rules for pheromone trail update: local and global. The local update rule is achieved during the ants' search and solution construction (i and j), according to Eq. 6. This update increases the pheromone on visited edges to dissuade old ones.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0$$

where 0 <ρ< 1 is the evaporation rate parameter. τ0 is the initial amount of pheromone trail.

ACS has a memorizing mechanism that allows it memorize the best solution among the solutions found so far compared with the solution found in the current iteration. Therefore, the global update rule is achieved by the best ACS memorized solution

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}$$

In which Δτij is defined follows:

$$\Delta\tau_{ij} = \begin{cases} 1/L_{gb}, & if\ arc\ (i,j) \in the\ best\ tour \\ 0, & otherwise, \end{cases}$$

where Lgb is the global best solution length.

The ACS algorithm described above is the general form used for a combinatorial optimization algorithm

This balance in the search space is the main factor in intelligently enhancing the solutions' quality and avoiding premature convergence problem. In this work, to solve the CSC problem, we proposed a disturbed process for ACS algorithm based on multi-agent.

Multi-Agent ACS for QOS-Aware CSC Problem

The high complexity of the CSC problem makes it hard to search in a reasonable time about optimal solution in most cases. The proposed multi-agent ACS (MAACS) method can reduce the problem complexity using searching decomposition and real-time population re-disruption.

The multi-agent system is widely used in distributed and parallel applications where it is composed of autonomous agents cooperating to simultaneously reach common objectives. Also, it introduces important capability to solve optimization problems using metaheuristic algorithms where it allows one agent to locally solve a sub-optimization problem with the purpose of pursuing to find global optimum by intersecting with other agents. In this work, we exploit this

capability to develop a multi-agent algorithm based on ACS for CSC problem, including a diffident agent with a special purpose, namely, MAACS.

In this section, present the architecture of the proposed algorithm. In this architecture, target a better architecture for MAACS in terms of the agents' number, agents' role, and maintaining good tradeoff between exploration and exploitation mechanisms.

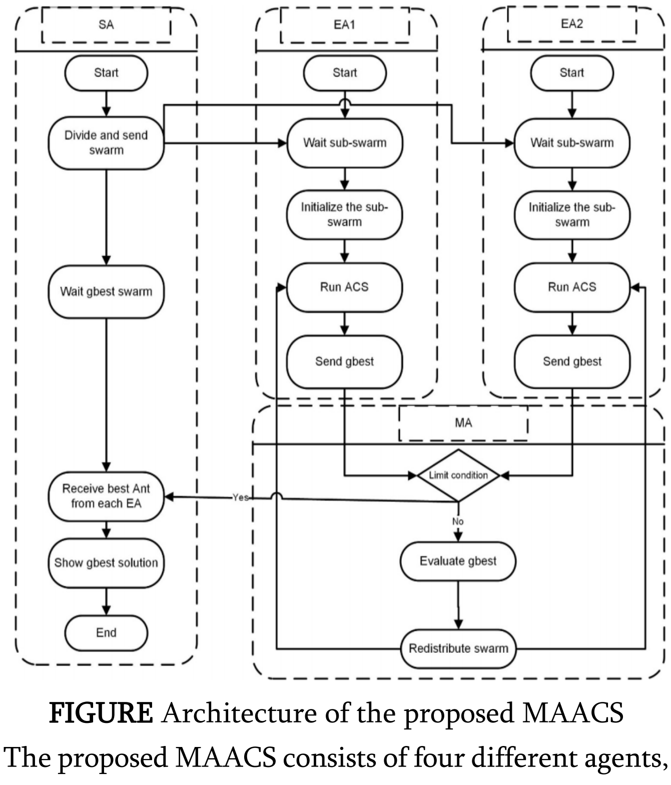


FIGURE Architecture of the proposed MAACS

The proposed MAACS consists of four different agents, namely, one synchronization agent (SA), two execution agents (EAs), and one monitorization agent (MA); each one is developed to do special action. The architecture of the proposed MAACS is shown in Figure, and the following sections describe the role of each agent

Synchronization Agent (SA)

This agent is composed of its acquaintances of all agents in the system and has different responsibilities. It is responsible for dividing the population into two equal parts and then for sending these parts to the EAs. Also, it is responsible for triggering the MA to receive the best solution of EAs in case of reaching the limit condition. Finally, it is responsible for determining the global solution between the different EAs' best solutions.

Execution Agents (EAs)

This agent executes the optimization process of the ACS algorithm. There are two different EAs, and each agent has a different initialization method. The first agent (EA1) randomly initializes the subpopulation, whereas the second agent (EA2) greedily initializes the subpopulation. After initializing the subpopulation, the ACS algorithm is executed and simultaneously, the best solution found so far by each agent is transmitted to the MA.

The EA agents construct one service at a time that represents a piece of the solution path. For each ant, It starts from task 1 (T1) and selects a web services CSj11 from the candidate services of T1 to be added into the solution path based on the stepwise transition rule in Eq.4. And then moves to the next task 2 (T2) and selects a new web service CSj12 and so on. The transition rule constructs the solutions incrementally using both the historical information which represents the amount of pheromone laid by the ants, and the heuristic information which represents the summation of the values of the QoS properties N (qos_r) where N (qos_r) has the same meaning as in Eq. This construction process continues until reaching the last task 2 (T2) and select a web serviceCS_1^{j2}. For constructed solutions, the ants update the pheromone trail using the local pheromone update rule in Eq. At the end of the solution construction, the best solution found thus far is selected by the best candidate ant, so this best candidate updates the pheromone on the best solution found thus far using a global update rule.

In each iteration, both EA agents constructs a solution and keep the best solution found thus far named gbest (as shown in Figure) and send throw the MA agent to the SA agent in case of reaching limit condition.

Monitorization Agent (MA)

This agent is composed of its acquaintances of EAs in the system. It monitors the performance of EAs and redistributes the population based on this performance. In the SA, the population is equally divided into two parts and is then sent to the EAs. However, in MA, it biases to a good EA that find better solution where it divides the population into two unequal parts by decreasing the population size of the bad EA solution to encourage the exploitation of the good EA agent. This limit condition determines the end of the system execution and returns the EAs' solutions to the SA agent to be compared and shown as the system's final solution.

Proposed System

System model and problem formulation

Consider a downlink of multiple TDMA systems in wireless networks each of which consists of one access point(AP) and users as illustrated in Fig. The set of systems is defined as K = {1, . . . , K}, where K is the number of systems. The set of users in system k is defined as Nk ={1, . . . , Nk}, where Nk is the number of users in system k. In system k ∈ K, its AP serves Nk users over discrete time horizon t ∈ {1, 2, . . .}. We assume that the wireless channels between the AP and users in each system are time-varying, but unchanged during a timeslot, and satisfy the Markov property, which are widely accepted.

In timeslot t, the AP in system k schedules one user ntk ∈ Nk and transmission power pk t ∈ P, where P is the set of candidate transmission power levels. A state of system k in timeslot t is defined as st k ∈ Sk, where Sk is a state space of system k. The state represents the feature information of each user such as channel gain and degree of QoS unsatisfaction. For simple notation, we denote user n in system k by a tuple (k, n). We denote the lth feature information of user (k, n) in timeslot t by fkt,n,l. An action of system k in timeslot t is defined by ak t = (ntk, pk t) ∈ Ak to indicate a scheduling decision. We denote a policy of system k by πk : Sk → Ak. Then, the instantaneous utility of system k in timeslot t is given by u(sk t, πk(sk t)), where u(·, ·) is a utility function commonly used in the systems. With the above ingredients, we can define a general resource allocation problem in wireless networks with multiple systems to maximize the total utility as follows:

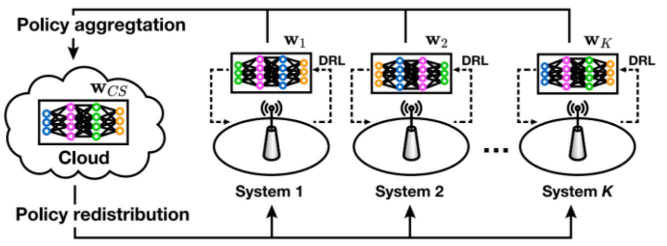


Fig. System architecture.

$$\underset{\pi}{\text{maximize}} \sum_{k\in\mathcal{K}} \mathbb{E}\left[\sum_{t=0}^{\infty} (\gamma)^t u(s_k^t, \pi_k(s_k^t))\right]$$

Where 0 < γ < 1 is a discount factor and π : ∏k∈K Sk → ∏k∈K Ak is a policy for all systems. For instance, a problem can be formulated to maximize the total average data rate by setting u(sk t, πk(sk t)) = r(sk t, πk(sk t)), where r(·, ·) is a function to calculate the instantaneous data rate.

FL for resource allocation

Can solve the problem in above equation by directly finding the (sub)optimal policy, π∗, for all the multiple systems. However, when the number of systems, K , is large, its complexity is too large to address. To resolve this issue, we decompose the problem according to each system k as

$$\underset{\pi_k}{\text{maximize}}\ \mathbb{E}\left[\sum_{t=0}^{\infty} (\gamma)^t u(s_k^t, \pi_k(s_k^t))\right]$$

Then, we can solve the problem in above equation by finding the (sub)optimal policy, $\pi k*$, for the decomposed problem for system k in a distributed manner. It is worth noting that the decomposed problem is a Markov decision process whose reward function and environment are given by the utility function $u(\cdot, \cdot)$ and the transition probabilities over state and action spaces, respectively. Hence, to solve the decomposed problem for each system, well-known DRL can be used as in the recent literature of resource allocation in wireless networks. In specific, each system k plays a role of an agent in DRL to learn the optimal policy $\pi k *$ to its decomposed problem.

The decomposed problems belong to the same class of problems for an identical task (i.e., resource allocation) with common utility functions. Hence, we may apply FL to solve them more efficiently. Specifically, if there exists a common policy π ¯ that can be used in any system, FL can be applied to learn it as in above Fig. In FL, each system individually learns its common policy via DRL approaches. Then, a cloud server aggregates the common policies from the systems and redistributes the aggregated one. This accelerates the learning speed of the common policy since it utilizes the experiences from all systems. Besides, it can provide adaptability to newly-arrived systems due to establishing new AP in wireless networks thanks to the common policy in the cloud server.

FL for resource allocation in wireless networks with multiple systems

Resource allocation policy structure for FL

The decomposed resource allocation problem for each system in above equation can be solved by learning the resource allocation policy via well-known DRL as in the recent literature. However, we cannot simply apply FL to aggregate the policy of each system since the structures of the policies depend on the system-specific characteristics. In specific, in most recent works, the state of system k in timeslot t is defined to directly provide the feature information as sk t =(fkt,1,1, . . . , fkt,1,L, . . . , fkt,Nk,L) ∈ Sk ⊆ RNk×L. Thus, the state space of system k, Sk, depends on the number of users in each system. Besides, the action space of system k also depends on it as Ak = Nk × P. These dependencies make the policy structures of systems different from each other. Consequently, applying FL for the policies becomes infeasible.

To resolve this issue and enable FL, we need a policy structure that fits any system (i.e., the common policy). To this end, we borrow a concept of circumstance-independent (CI) policy structure. Instead of directly describing the states and actions as described above, it represents the states as whether a user in a specific condition of the state information exists or not in the system. In addition, it chooses the scheduling decision as one of the conditions. With the CI-policy structure, we can represent the states and actions regardless of the change of network circumstances. We refer the readers for more detailed description.

The CI-policy structure is originally proposed to address the dynamic changes of the circumstance in a single system. However, its notion can be adopted to generalize the policies for multiple systems as well. To this end, here we define a common state space S and the common action space $\bar{A}$ that can describe any states and actions in different systems as illustrated in below Fig. The common state and action spaces can be constructed according to the definitions of states and actions in the CI policy. We then define translation functions system k, ds k (·) and dk a(·), that map Sk to $\bar{S}^-$ and $\bar{A}$ to Ak, respectively. By using them, in timeslot t, we can translate any state sk t ∈ Sk into $\bar{s}$ t ∈ $\bar{S}$ and any action a t ∈ A according to given state $\bar{s}$ t ∈ $\bar{S}$ into at k ∈ Ak as illustrated in Fig. 2. Hence, a common policy π : S → A with translation functions can play a key role of a universal interpreter to apply FL for resource allocation of multiple systems.

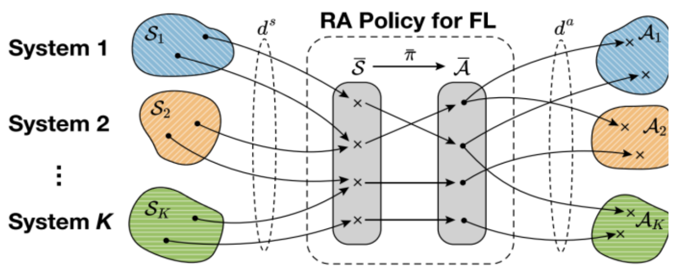
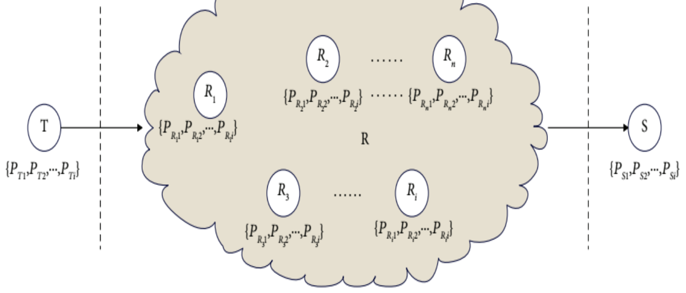


Fig. Illustration of the proposed policy structure for FL

Dynamic Resource Allocation Algorithm

Task Service-Oriented Network Resource Management Model of Space-Earth Integration. In the process that the network provides services for the task, the initial starting state is abstracted as a virtual node and defined as the source node T, and the logical termination state after reaching the task goal is abstracted as the sink node S. TRS model logic network diagram is shown in Figure. At the source node, we use $T = \{Ti|i = 1, 2, . . . , n\}$ to describe the defined set of network functions and use $A_T = \{a_{T1}, a_{T2}, . . . , a_{Ti}\}$ to describe the set of network functions required by the task.

Among them, $a_{Ti} = 0/1$ indicates whether the task has requirements for a defined network function. We use $P_T = \{p_{T1}, p_{T2}, . . . , p_{Ti}\}$ to describe the performance requirements that the network function needs to achieve, and use to describe the constraints on the task. At the sink node, we use $S = \{Si|i = 1, 2, . . . , n\}$ to describe the service performance parameters (such as delay and processing cost) of different tasks, and use $P_s = \{p_{S1}, p_{S2}, . . . , p_{Si}\}$ to describe the service performance. Among them, is the service performance metric vector corresponding to the network function Ti required by the task.


T
$\{P_{T1}, P_{T2}, \cdots, P_{Ti}\}$
R_1
R_2
R_n
R_3
R_i
R
S
$\{P_{S1}, P_{S2}, \cdots, P_{Si}\}$

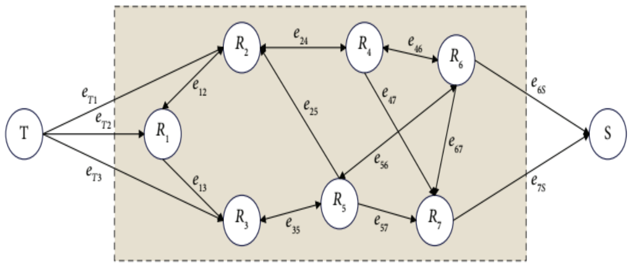


Figure TRS model logic network diagram.

In the resource pool, use $R = \{R_i|i = 1, 2, . . . , n\}$ to describe the node set in the resource pool, where n is the number of resource nodes, and use $r = \{ri|i = 1, 2, . . . , n\}$ to describe the different virtual resource types in the resource pool. We use $A_R = \{a_{r1}, a_{r2}, . . . , a_{ri}\}$ to describe the resource capability, where $a_{ri} = 0/1$ represents whether it is capable of a certain virtual resource ri (such as storage and computing) defined by abstraction. We use $P_R = \{p_{r1}, p_{r2}, . . . , p_{ri}\}$ to describe the performance index of resource capability, where is the vector of the performance index measure corresponding to resource ri capability; $C_R = \{c_{r1}, c_{r2}, . . . , c_{ri}\}$ is used to describe the resource constraints. In addition to node resources, another important resource in the resource pool is link resources. The logical connectable relationship between resource nodes is based on the physical connectable relationship of resource entities. Three logical links are defined in the TRS model. The connection between the virtual source node and the network resource entity node is defined as the source link, which is represented by e_{Ti}. The connection relationship between resource nodes and is defined as a node link, which is represented by e_{Ti}. The connection between the network resource entity node and the virtual sink node defines the sink link, which is represented by e_{T_j} . Among them, there can be multiple source links and sink links, which are determined according to specific mission objectives and constraints.

Hierarchical Graph Model Based on TRS.

The advantage of using the TRS model is that the problem of solving the SFC mapping scheme is converted into the shortest path search problem in graph theory, and then, the existing algorithm can be used to solve it conveniently. The specific processing process of the hierarchical graph model is introduced below and is described in conjunction with the example given in Figure

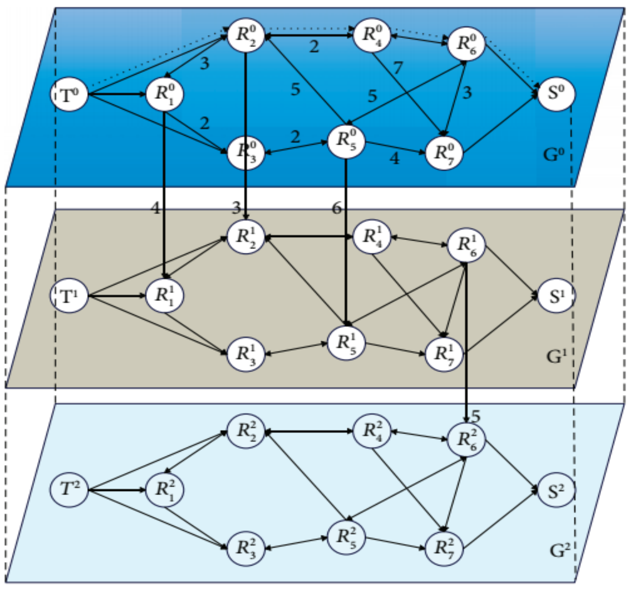


Fig An example of a hierarchical graph model based on TRS.

Since $T = \{T_i | i = 1, 2, \ldots, m\}$ is used to describe the defined network function set in the TRS model, it can be used to represent all the defined VNF sets in the NFV environment. SFC is composed of different VNFs connected in a certain order. We define to $\text{req} = \{n_1, n_2, \ldots, n_k\}$ describe the sequence of service functions requested by SFC so that SFC and network function set T can correspond, that is, $\text{sfc} = \{T_{n_1}, T_{n_2}, \ldots, T_{n_k}\}$. Among them, req is a set of integer sequence values, and the numerical size satisfies $1 \leq (\forall \quad n_i \in \text{req}) \leq m$. SFC is a logical chain of virtual ordered connections, and it is necessary to map each virtual network function and virtual link to the appropriate resource nodes and transmission links in the resource pool. We define this mapping relationship as Mapping= $\{M_{\text{node}\ (Tr_i)}, M_{\text{path}\ (Tr_i Tr_{i+1})}\}$, where $M_{node(Tr_i)}$ represents the set of mappable resource nodes of the i-th VNF , and represents the set of mappable paths between the two VNFs and $M_{path}(Tr_i, Tr_{i+1})$. The process of generating the final solution will need to select from these two sets according to a certain strategy.

In the example in Figure, T = {T1, T2, T3, T4}, req={4, 1} is given, then sf = {T4, T1}, assume that the resource nodes that T4 can map are R1, R2, and R3. The resource node that T1 can map is R6, and the cost value of each link is marked in the base layer. The analysis shows that the optional paths of SFC are T0 ⟶ R0 1 ⟶ R1 1 ⟶ R1 6 ⟶ R2 6 ⟶ S2, T0 ⟶ R0 2 ⟶ R1 2 ⟶ R1 6 ⟶ R2 6 ⟶ S2 and T0 ⟶ R0 5 ⟶ R1 5 ⟶ R1 6 ⟶ R2 6 ⟶ S2. Among them, the relationship between the two path nodes may not be directly reachable. For example, there are two options T ⟶ R 2 ⟶ R5 and T ⟶ R1 ⟶ R3 ⟶ R5 for T ⟶ R5. Different path choices will have different path cost values, and it calculates the minimum cost value of the SCC optional path, which are 18, 13, and 20, respectively.

Therefore, the path selected across layers is T0 ⟶ R0 2 ⟶ R1 2 ⟶ R1 4 ⟶ R1 6 ⟶ R2 6 ⟶ S2, and the path corresponding to the actual selection in the base layer is T ⟶ R 2 ⟶ R4 ⟶ R6 ⟶ S.

End-to-End Cross-Domain Collaborative Resource Management Model

In the NFV environment at the centralized control and coordination nodes of the network architecture, we define a higher level of abstraction VNF to construct a simplified main SFC and then divide the main SFC into sub-SFCs that can be executed internally by different resource domains. It formulates a globally optimal SFC mapping path by feeding back calculation results and mapping result information in each resource domain. The resource management model of end-to-end cross-domain collaboration will be introduced in detail below, as shown in Figure.

use MSFC to represent the main SFC, and SSFC to represent the sub-SFC. An MSFC is composed of multiple SSFCs, which can be expressed as MSFC = {SSFCi|i= 1, 2, . . . , n}. The same SSFC can be supported by different resource domains, and there can be multiple schemes to meet the requirements for the same SSFC in one resource domain.

Cross-Domain Collaborative Resource Allocation Method Based on TRS Hierarchical Graph Model. In the resource domain, the TRS model is used to construct the task serviceoriented network resource topology, which can be expressed as G = (R, E, T, S). *e resource node with service capability in the hybrid NFV network environment is denoted as RS, the resource node set that can support VNF instantiation is denoted as RVNF, and the traditional physical device node set is denoted by RPNF. In the way of resource sharing, it is considered that a resource node can carry multiple SFs, which will be processed one by one in a queue-based manner internally.

$$R = R^S = R^{VNF} \cup R^{PNF} = \{R_1, R_2, \ldots, R_n\}.$$

We define the resource type set as an enumeration type, and the elements in the set r represent computing resources, transmission resources, storage resources, sensing resources, and navigation resources, respectively.

$$r = \{r_{cpu}, r_{tran}, r_{\text{stor}}, r_{\text{sens}}, r_{\text{navi}}\}.$$

Parametric characterization was performed using the TRS model. At the resource node, a set of binary variables Ari is used to indicate whether there is a corresponding type of resource capability, and a set of values is used to represent the size of the available resources of the corresponding type, and a set C_{ri} is used to represent the constraints of the corresponding resource capabilities. We define the attribute parameters of resource constraints of two nodes, where uc(r_i) represents the usage cost of resource r_i per unit resource, and ut(r_i) represents the processing delay per unit resource of resource r_i.

$$A_{R_i} = \{a_{r_i} | r_i \in r\},$$
$$P_{R_i} = \{p_{r_i} | r_i \in r\},$$
$$C_{R_i} = \{c_{r_i} | r_i \in r\} = \{\{\text{uc}(r_i), \text{ut}(r_i)\} | r_i \in r\}.$$

By calculating Ari and P_{Ri}, the available resource capacity of the entire resource pool can be obtained, which is represented by R^{total}, and the five values in the set represent the total amount of each type of available resources in the resource pool.

$$R^{\text{total}} = \sum_{i=1}^{n} P_{R_i}, = \{r_{cpu}^{\text{total}}, r_{\text{tran}}^{\text{total}}, r_{\text{stor}}^{\text{total}}, r_{\text{sens}}^{\text{total}}, r_{\text{navi}}^{\text{total}}\}.$$

The link set E consists of the source link set, the node link set, and the sink link set defined in the TRS model. The adjacency matrix ER is used to represent the existence of node links and the delay cost of link establishment, where n represents the number of resource nodes in the domain and e_{ij} represents the weight of delay cost for link establishment. If i= j, then e_{ij} = 0. If e_{ij}= ∞, it means that the two resource nodes cannot communicate directly. If eij= w, w ∈ R+(R+represents a positive real number), it means that resource nodes R_i and R_j can communicate, and the delay cost of establishing a link is w. In the static graph model, in the same time slice, only the delay cost consumed when the link is established for the first time needs to be 1 when the link is established at the end of the link; otherwise, the value is 0

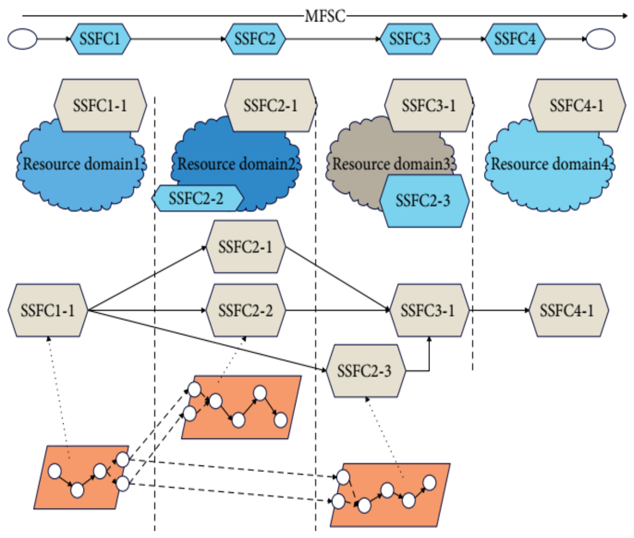


Figure Example of cross-domain collaborative resource management model.

$$E^R = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nn} \end{bmatrix}.$$

Assume that the definitions of network service interfaces and related network functions follow the existing relevant standards, and here, focus more on solving the abstracted SF orchestration problem. At the source node, the m SFs that can be supported are represented by a set T, and the defined SFs are unique and nonrepetitive.

$$T = \{T_1, T_2, \ldots, T_m\}.$$

The SFC request is represented by SFC, which emphasizes the order in which the traffic passes through the SF, and it allows the repetition of SFs that appear before and after in the chain. We associate SFC with elements in the network function set T by defining a set of integer sequence values req to represent the SF order requested by SFC. Ten, T_{n1} represents the virtual SF node in the service chain, and the logical link set of an SFC is represented as E^{sfc} A set C^{sfc} is used to describe the constraints on SFC, and two constraint attribute parameters are defined, where dc(Ti) represents the upper limit of resource cost acceptable to T_i by the task, and dt(T_i) represents the upper limit of resource processing time acceptable to Ti by the task.

$$\text{req} = \{n_1, n_2, \ldots, n_k\}, 1 \le n_i \le m,$$

$$\text{SFC} = \{T_{n_1}, T_{n_2}, \ldots, T_{n_k}\},$$

$$E^{sfc} = \{(T_{n_1}, T_{n_2}), (T_{n_2}, T_{n_3}), \ldots, (T_{n_{k-1}}, T_{n_k})\},$$

$$C_{sfc} = \left\{c_{T_{n_i}} | n_i \in \text{req}\right\} = \{\{dc(T_i), dt(T_i)\} | n_i \in \text{req}\}.$$

The set P^{sfc} is used to describe the various resources required by each SF in SFC to meet the performance requirements, and it is composed of the numerical value of the total amount.

$$P_{sfc} = \left\{p_{T_{n_i}} | n_i \in \text{req}\right\},$$
$$= \left\{\{dr^i_{cpu}, dr^i_{\text{tran}}, dr^i_{\text{stor}}, dr^i_{\text{sens}}, dr^i_{\text{navi}}\} | 1 \le i \le k\right\},$$

$$d\_R^{\text{total}} = \sum_{i=1}^{k} p_{T_{n_i}},$$
$$= \left\{d\_r^{\text{total}}_{\text{cpu}}, d\_r^{\text{total}}_{\text{tran}}, d\_r^{\text{total}}_{\text{stor}}, d\_r^{\text{total}}_{\text{sens}}, d\_r^{\text{total}}_{\text{navi}}\right\}.$$

The resource allocation in the domain is solved by constructing a hierarchical graph model, and the purpose is to find the mapping scheme of the virtual nodes and virtual links of SFC. The task-oriented service-oriented network resource topology G constructed by using the TRS model is used as the base layer G^0, and the m layer is copied and extended to form a hierarchical graph model G^ω

$$G^\omega = \{G^0, G^1, G^2, \ldots, G^m\}.$$

By judging whether the resource margin of the resource node is sufficient, and at the same time judging whether the resource cost and the delay cost are less than the upper limit of the task requirement, it is determined whether T_{ni} can be mapped to the resource node Rj, and the interlayer link of Gi– 1 and Gi is used to represent the resource node that TnI can be mapped to. We define the operation of multiplying the corresponding elements of two sets of the same length as ⊗ and use Rij to represent the resource node Rj of the i-th layer of the hierarchical graph. For the resource domain related to satellite network, the mapping rule is adjusted, and the probability of mapping to the existing link is moderately increased, so as to reduce the increased mapping delay cost caused by waiting for the antenna deflection to establish the link. Specifically, it can be realized by changing the logical weight of the link.

$$T_{n_i} \longrightarrow R_j^i: \begin{cases} a_{T_i} \cdot p_{T_i} < a_{R_j} \cdot p_{R_j}, \\ p_{T_{n_i}} \otimes uc(r_j) < dc(T_{n_i}), r \\ p_{T_{n_i}} \otimes ut(r_j) < dt(T_{n_i}). \end{cases}$$

If the SFC arrival time of a service request is represented by t_s, and the time to complete the last SF mapping of the service request is rep resented by t_e, then the service time t_c is defined as the time difference between completion and arrival. The length of service time is related to delay, which can be used to measure service quality. The deadline for service requests is denoted by tl. The deadline is the final time limit for completing a given service mapping, beyond which the service request is considered to have failed. In addition, the deadline can also reflect the priority of the task; the shorter the t l, the higher the priority.

$$t_c = t_e - t_s \leq t_l.$$

The different paths selected across layers are mapped to G^0 to form a candidate scheme, which is denoted by SSFC. The scheme includes the selected resource node set SR and the selected link set SE. It is also necessary to find the boundary node information that the resource nodes mapped by the SF at the end of each candidate scheme can be connected, and form a set SB and store it in the candidate scheme set. For the nodes in the domain, the border node information of the domain can be obtained through the Border Gateway Protocol.

$$\text{ssfc} = \{SR, SE, SB\}$$

After completing the formulation of the mapping scheme, since the TRS model abstracts the logical termination state after reaching the task goal as the sink node S, the service performance metric vector set P_{ssfc} can be used to describe the service performance achieved according to the mapping scheme ssfc. The measurement of the parameters of the service performance includes two points: the resource cost of the intra domain mapping scheme and the delay cost.

$$P_{ssfc} = \{\text{cost}_{ssfc}, \text{delay}_{ssfc}\}.$$

We denote the cost overhead required by the selected resource node to process SFC by CD. Since Tni has different demands for different types of resources, different types of resources have different costs uc(r). Then, through the operations defined in the curly brackets of the following formula, the multidimensional resource cost will be a numerical set with five elements, and the sum of this numerical set is used as the size of the node cost CD. Among them, for

the elements of the two sets A and B of equal length, the operation of dividing the elements in the set is defined as A|B; cij is a binary variable. If the Tni in the sfc is mapped to the resource node R j, the value is 1; otherwise, it is 0.

$$\text{CD} = \sum \left\{ \sum_{i=1}^{k} \sum_{j=1}^{n} \gamma_{ij} \cdot p_{T_{n_i}} \otimes uc(r_j) | d\_R^{\text{total}} \right\}, n_i \in \text{req}, r_j \in r.$$

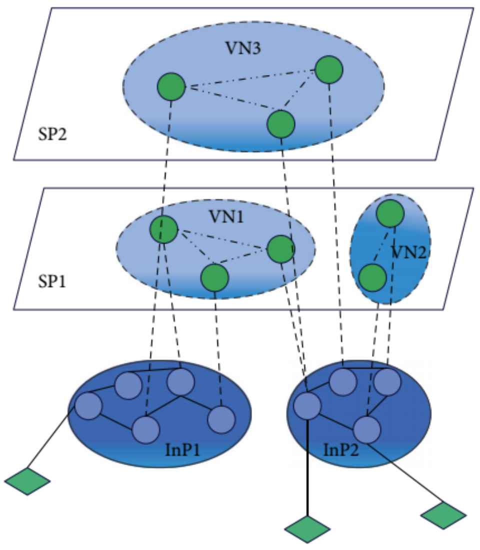


Figure Virtual network mapping model

We use CF to denote the total delay cost of the intra domain mapping scheme, including node delay and link delay.

$$CF = t_{\text{node}} + t_{\text{link}}$$

Node delay refers to the sum of the processing time spent on mapping all virtual nodes in sfc to resource nodes, which is represented by t_{node}. Since a T_{ni} in sfc has different requirements for each type of resource, each value in the multidimensional resource processing delay cost set is not necessarily the same when combined with the unit processing delay cost. We take the maximum value as the time spent by the Tni-map node to process T_{ni}, and further sum up to get t_{node}.

$$t_{\text{node}} = \sum_{i=1}^{k}\sum_{j=1}^{n} \max\left\{\gamma_{ij} \cdot p_{T_{n_i}} \otimes ut\left(r_j\right)\right\},$$

$$n_i \in \text{req}, r_j \in r.$$

The delay spent on the link includes link establishment delay, transmission delay, and propagation delay, which is represented by t_{link}. Among them, ps represent the size of the transmitted data packet, and bwe represents the bandwidth capacity of the link. The link propagation delay is obtained by dividing the link length by the signal propagation rate in the communication medium, which is denoted by tpe. The bandwidth and propagation delay of the link are known data information in advance as attributes of the link.

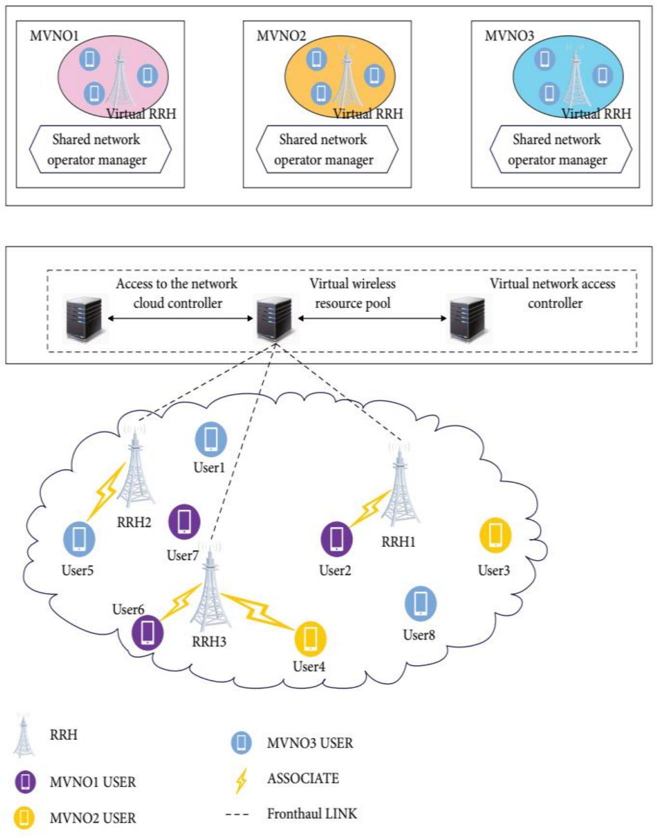

MVNO1
Virtual RRH
Shared network operator manager
MVNO2
Virtual RRH
Shared network operator manager
MVNO3
Virtual RRH
Shared network operator manager
Access to the network cloud controller
Virtual wireless resource pool
Virtual network access controller
User1
RRH2
User7
RRH1
User5
User2
User3
User6
RRH3
User8
User4
RRH
MVNO3 USER
MVNO1 USER
ASSOCIATE
MVNO2 USER
Fronthaul LINK

Figure WNV-CRAN architecture.

$$t_{link} = \sum_{e_{ij} \in SE} \left(sta_{e_{ij}} \cdot e_{ij} + \frac{ps}{bw_{e_{ij}}} + tp_{e_{ij}} \right)$$

After the calculation is completed in different resource domains, the intradomain mapping schemes generated for different SSFCs will be returned. The master MANO at the centralized coordination and control node will be responsible for labeling the domain information of the received candidate solutions and store the corresponding intradomain mapping solutions in the corresponding set SSFCi. SSFC i is the virtual node in MSFC, (SSFCi, SSFCi+1) represents the directed virtual link from SSFCi to SSFCi+1, and EMSFC represents the virtual link set of MSFC.

$$\text{MSFC} = \left\{\text{SSFC}_1, \text{SSFC}_2, \ldots, \text{SSFC}_p\right\} = \{\text{SSFCi} | 1 \le i \le p\},$$

$$E^{MSFC} = \left\{\left(\text{SSFC}_1, \text{SSFC}_2\right), \left(\text{SSFC}_2, \text{SSFC}_3\right), \ldots \left(\text{SSFC}_{p-1}, \text{SSFC}_p\right)\right\}.$$

In the multiresource domain network environment, the multidomain network topology consists of the topology of each resource domain and the interdomain links. However, due to the limited domain visibility, the centralized coordination control node cannot know the topology of each resource domain in advance, and it only has information about the boundary nodes and interdomain links of each domain. For a multidomain network with k resource domains, there are

$$G^{global} = G_1 \cup G_2 \cup \cdots \cup G_k \cup E^L$$

The main MANO will set the weight allocation ratio in the optimization target according to the QoS requirements of the service request. It uses the different resource cost 23$_{stssfc}$ and delay cost delay$_{ssfc}$ of each mapping scheme, as well as the stored interdomain link information for calculation, and denotes the final selected cross-domain cooperative mapping scheme as M.

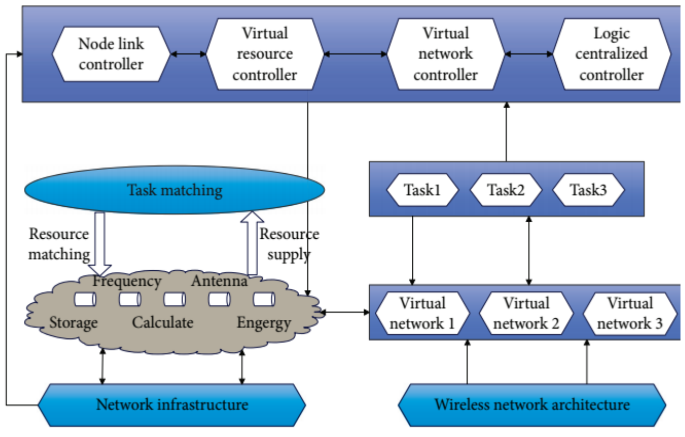


Figure Network resource virtualization.

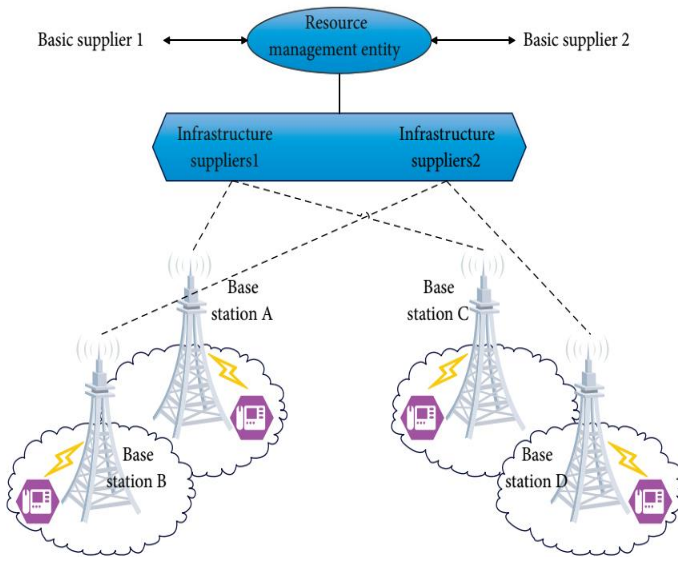


Figure A network architecture for service providers and infrastructure providers to virtualize wireless networks

The interdomain link formed by border nodes is a subset of EL, and its delay includes two parts: the transmission delay and the propagation delay of the link, and its specific calculation method is similar to the calculation of the intradomain link delay.

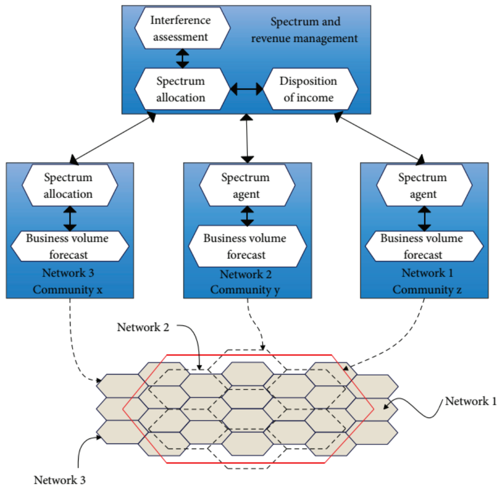
$$\text{delay} = \sum_{ssfc \in M} \text{delay}_{ssfc} + \sum_{e_{ij} \in M} \left(\frac{ps}{bw_{e_{ij}}} + tp_{e_{ij}} \right)$$

We use cost to denote the end-to-end resource processing cost of MSFC. Since it can be considered that the boundary node is mainly responsible for the forwarding function, the resource processing cost of the boundary can be approximately ignored. Then, the cost of the MSFC end-to-end mapping scheme will be the sum of the resource costs of the mapping schemes in each SSFC domain selected in M.

$$\text{cost} = \sum_{ssfc \in M} \text{cost}_{ssfc}$$

The interdomain resource mapping allocation scheme solved by the CRAM-AMD method, and its purpose is to provide end-to-end network services for users. For the measure of the pros and cons of the scheme, it can be considered as the comprehensive cost of the linear combination of the above two index parameters delay and cost, which is defined as CMSFC. Generally speaking, low latency network service means higher resource cost overhead, which means that these two parts of the optimization goal are in conflict with each other to a certain extent. It needs to weigh the proportion of allocation according to the needs of specific scenarios and use δ and μ to coordinate the proportion of the two parts of the optimization goal.

$$min\, C_{MSFC} = min(\delta . delay + \mu . \cos t)$$



(a)

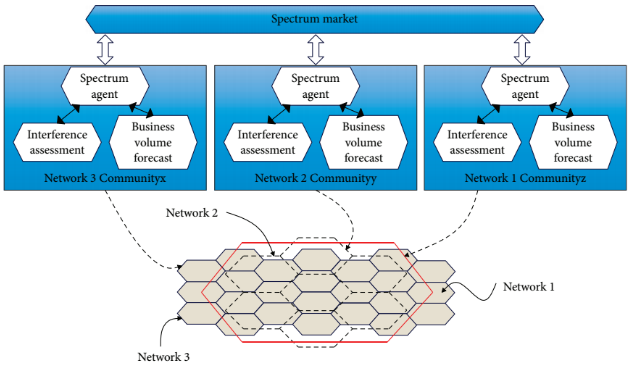
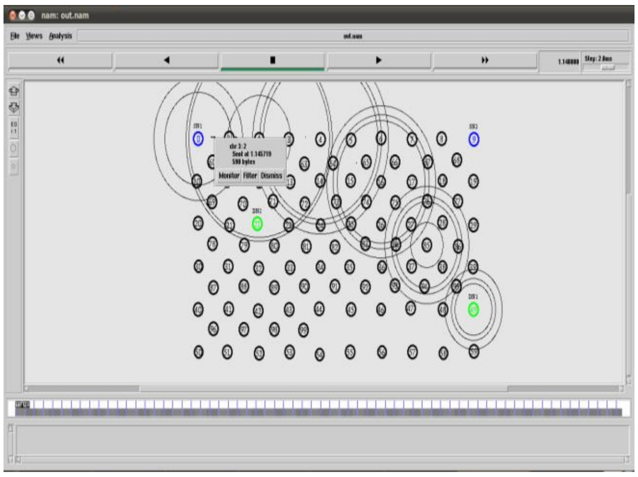
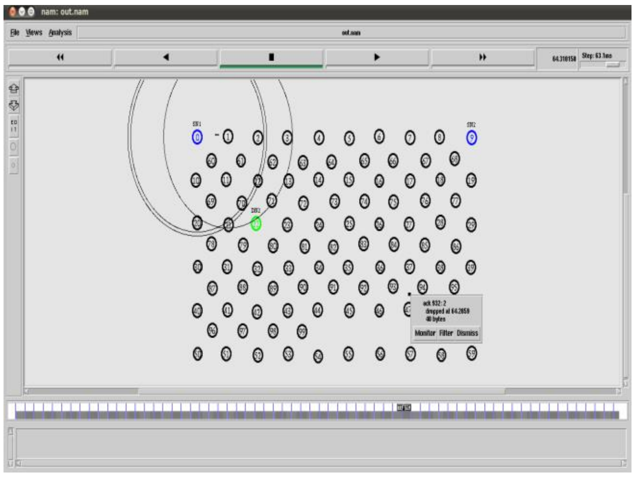
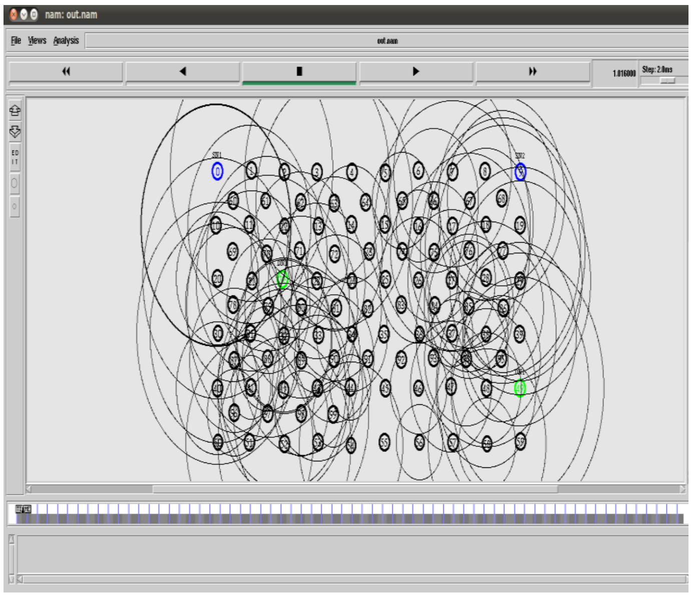


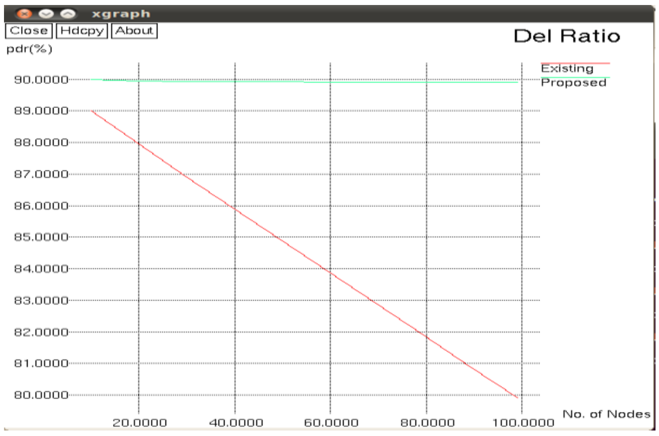
Figure Dynamic spectrum management. (a) Dynamic spectrum management of centralized networks. (b) Dynamic spectrum management of distributed networks

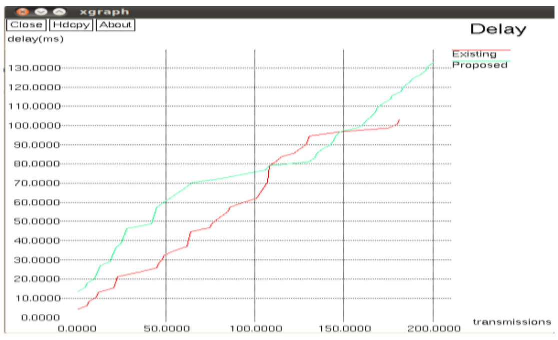
Screenshot

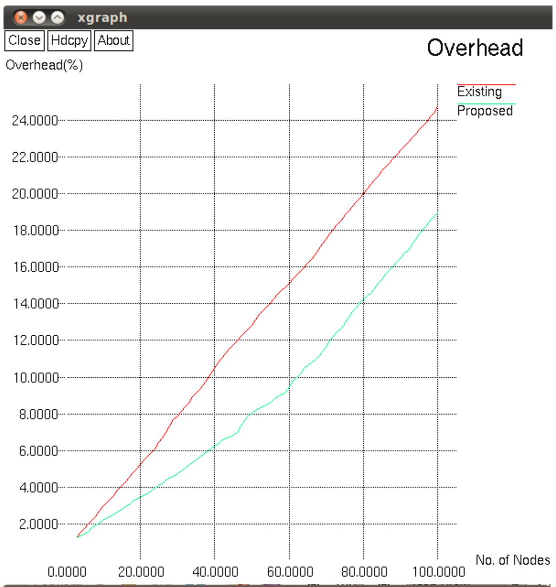


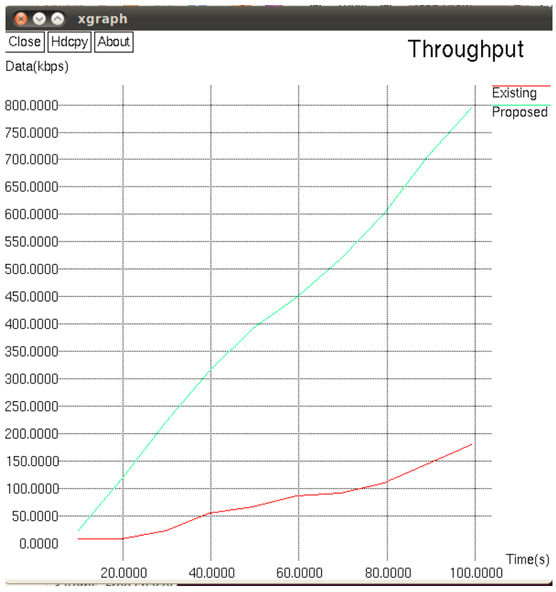












System Requirements

Software Requirements

Operating System	: Windows 7/8/10
Software package	: Visual Studio 2010
Front End	: ASP.NET, C#
Back End	: SQL Server 2008

Hardware Requirements

Processor	: Dual Core 2.0 Ghz (min)
Hard Disk Space	: 10 Gb
Physical Memory	: 2 Gb
Input	: Mouse and Keyboard
Output	: Color Monitor

V. CONCLUSION

For network connections, link and node resources are an important basis for providing reliable guarantees, and virtualizing link and node resources can usually effectively improve network performance. The mapping algorithm of link and node sum is often the focus of wireless network virtualization. The reason is that the link bandwidth of the wireless link and the node capacity is limited. Therefore, when deploying wireless network link and node mapping, it is necessary to reasonably increase the link bandwidth and node capacity constraints. This paper combines the dynamic resource allocation algorithm to construct a wireless network virtualization resource sharing model to improve the efficiency of wireless network operation. Through comparative research, it can be seen that the wireless network virtualization resource sharing method proposed in this work considering the dynamic resource allocation algorithm can effectively improve the processing efficiency of wireless network virtualization resources. As a future work on this subject, an extension of the proposed dynamic framework into addressing interference issues and allocation failures can be considered.

VI. REFERENCES

[1]. A Range-Based Secure Localization Algorithm for Wireless Sensor Networks Xingcheng Liu, Senior Member, IEEE, Shaohua Su, Feng Han, Yitong Liu, Zhihong Pan : DOI 10.1109/JSEN.2018.2877306, IEEE Sensors Journal

[2]. HybLoc: Hybrid Indoor Wi-Fi Localization using Soft Clustering based Random Decision Forest Ensembles Beenish A. Akram1 , Ali H. Akbar1 , and Omair Shafiq2 1Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan 2Carleton School of Information Technology, Carleton University, Ottawa, Ontario, Canada

Cite this article as :

C. Dina, A. Suriya, P. Madhavan, A. Alfred Iraianban, Alexander. A, "An Effective Multi-Agent Ant Colony Optimization Algorithm for QoS-Aware Cloud Service Composition", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 10 Issue 3, pp. 576-602, May-June 2023.
Available at doi :
https://doi.org/10.32628/IJSRSET23103171
Journal URL : https://ijsrset.com/IJSRSET23103171