2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

Research on Cooperative Scheduling Model of Computing Power and Network Resources

Changqing Wang [1,a]
[1]China Academy of Industrial Internet
Beijing, China
[a]wangchangqing@china-aii.com

Shuyan Yue [2,b]
[2]China Academy of Industrial Internet
Beijing, China
[b]yueshuyan@china-aii.com

Jiehao Chen [3,*]
[3]China Academy of Industrial Internet
Beijing, China
* Corresponding author: chenjiehao@china-aii.com

***Abstract*—This paper investigates the computing- network cooperative scheduling model, and discusses how to realize efficient cooperation between computing resources and network resources to improve the overall system performance. First, this paper analyzes the shortcomings of the existing scheduling strategies and points out the problems in resource utilization, scheduling efficiency and system stability. Second, an optimization model is proposed. The model is able to achieve balance between multiple performance indicators by introducing a multi-objective optimization algorithm. Experimental validation shows that the model effectively improves the system performance in multiple application scenarios. Finally, this paper summarizes the research results, provides a new idea and reference for the future cooperative scheduling strategy of computing-network, and looks forward to the future research direction.**

Keywords-Computing-Network coordination, Scheduling Strategy, Multi-Objective Optimization

I. INTRODUCTION

With the rise of emerging technologies such as cloud computing and edge computing, computing-network coordination, as a new type of resource management technology, effectively combines computing and network resources through intelligent scheduling and allocation, so as to improve resource utilization, reduce system latency, and achieve optimal utilization of resources[1][2]. In recent years, a variety of scheduling models for computing-network coordination have been proposed in academia and industry, such as linear programming models, scheduling models based on heuristic algorithms, and machine learning models[3][4]. However, linear programming models have high computational complexity when dealing with large-scale and dynamically changing network environments; heuristic algorithms rely on algorithmic parameter settings, which may lead to poor stability and consistency of the results; and machine learning models require a large amount of training data and computational resources, which limits their practical applications. In order to meet the complex and changing application scenarios, we propose a computing-network cooperative multi-objective optimization model that can dynamically adapt to the changing environment. During the scheduling process of computation and network collaboration, the model comprehensively considers multiple performance indicators such as computational resource utilization, network bandwidth utilization, task completion time, and energy consumption, and strives to seek the optimal solution among multiple objectives, thus improving the system efficiency and intelligence. In the optimization model, we develop a multi-objective optimization algorithm based on genetic algorithm and combine it with simulated annealing algorithm to effectively avoid falling into local extremes[5].

In summary, our main arguments are, firstly, to design a mathematical optimization model that comprehensively describes the computing-network coordination problem. The model includes an objective function and constraints, where the objective function focuses on maximizing the resource utilization and minimizing the service response time, while the constraints will ensure the feasibility of the model and the accuracy of the practical application. Second, an efficient solution algorithm is developed to solve the complex scheduling problem presented in the optimization model.

II. RELATED WORK

Computing-Network Coordination. By coordinating the cooperative scheduling and management of computing and network resources, it realizes the efficient distribution of computing tasks on different computing resources and ensures the fast transmission and processing of data in the network. Multiple factors need to be considered for computing-network coordination, including the load of computing resources, the availability of network bandwidth, and the prioritization of tasks. By dynamically adjusting the allocation of computing and network resources, the overall performance of the system can be improved while ensuring the task completion time[6].

Scheduling Strategies. Scheduling policies can be categorized into three types: static scheduling policies, dynamic scheduling policies, and hybrid scheduling policies[7]. Static scheduling policies allocate computational and network resources based on predefined rules and algorithms. Classical methods include shortest path first (SPF) and minimum cost routing (MCR), etc. Dynamic scheduling strategies allocate resources based on real-time network state and computational demand, and commonly used methods include dynamic load balancing (DLB) and real-time task scheduling (RTS). Hybrid scheduling strategy combines the advantages of static and dynamic scheduling and adopts a hierarchical and phased scheduling approach. Usually, a hybrid scheduling strategy will use static scheduling methods for approximate resource allocation in the initial stage, and then dynamically adjust according to the real-time situation during operation.

Optimization Model. Transform the resource allocation problem into a mathematical optimization problem, and use numerical solution methods to find the optimal resource allocation scheme. Current research mainly focuses on resource allocation, task scheduling and the improvement of network resource utilization efficiency, including linear programming, integer programming and heuristic algorithms[8][9]. Linear programming is often used to solve the resource allocation problem, and its advantage is that it can provide the global optimal solution. However, due to the complexity and variability of the computing- network coordination problem, linear programming has high computational complexity when dealing with large-scale and dynamically changing network environments. Integer programming provides more flexible optimization schemes with constraints, but also faces the challenges of long computational time and difficulty in solving. Heuristic algorithms, such as genetic algorithms and particle swarm optimization, are widely used in computing-network coordination due to their fast solution speed and good adaptability. These algorithms find approximate optimal solutions by simulating natural evolution or population behavior, and are particularly suitable for large-scale problems and nonlinear optimization problems. However, the results of heuristic algorithms depend on the setting of algorithmic parameters, which may lead to poor stability and consistency of solutions. Machine learning, on the other hand, can dynamically adjust the scheduling strategy and improve the system's adaptive capability, but it requires a large amount of training data and computational resources, which may be limited in practical applications.

III. MATHEMATICAL MODEL

A. *Model Description*

Let ($C = \{C_1, C_2, ..., C_m\}$) denote the set of computational nodes, ($N = \{N_1, N_2, ..., N_m\}$) denote the set of network nodes, and ($T = \{ T_1, T_2, ..., T_k \}$) denote the set of tasks. Each task T_i can be represented as a triple $T_i = (d_i, p_i, r_i)$, where d_i is the data size of the task, p_i is the computational requirement of the task, and r_i is the communication requirement of the task. The computational power of the computing node C_j is denoted by c_j, and the bandwidth of the network node N_l is denoted by b_l. x_{ij} denotes whether the task T_i is executed on the computing node C_j, and 1 if it is executed, and 0 otherwise. y_{il} denotes whether the task T_i communicates through the network node N_l, and 1 if it communicates, and 0 otherwise.

$$[min \sum_{i=1}^{k} (\sum_{j=1}^{m} x_{ij} \cdot \frac{p_i}{c_j} + \sum_{l=1}^{n} y_{il} \cdot \frac{r_i}{b_l})] \tag{1}$$

denotes the total task completion time, where the first term denotes the computation time of the task on the computation node and the second term denotes the communication time of the task on the network node. We assume that each task can only be executed on one compute node,

$$[\sum_{j=1}^{m} x_{ij} = 1, \forall i \in 1,2, \dots, k]. \tag{2}$$

Each task can only communicate through one network node,

$$[\sum_{l=1}^{n} y_{il} = 1, \forall i \in 1,2, \dots, k]. \tag{3}$$

The computing power of a computing node cannot exceed its capacity,

$$[\sum_{i=1}^{k} x_{ij} \cdot p_i \le c_j, \forall j \in 1,2, \dots, m]. \tag{4}$$

The bandwidth of a network node cannot exceed its capacity,

$$[\sum_{i=1}^{k} y_{il} \cdot r_i \le b_l, \forall l \in 1,2, \dots, n]. \tag{5}$$

B. *Objective function*

Maximize the network bandwidth utilization

$$\left[\text{Maximize} \sum_{i=1}^{N} B_i \times U_i\right], \tag{6}$$

where B_i represents the bandwidth used by the (i)th task, U_i represents the utilization of this bandwidth, and N is the total number of tasks. Minimize the execution time of the tasks

$$[\text{Minimize} \sum_{i=1}^{N} (T_i^{\text{compute}} + T_i^{\text{transmit}})], \tag{7}$$

where T_i^{compute} represents the computation time of the (i)th task and T_i^{transmit} represents the data transmission time of the task. Minimize the overall energy consumption of the system

$$[\text{Minimize} \sum_{i=1}^{N} E_i], \tag{8}$$

where E_i represents the energy consumed by the (i)th task during computation and transmission. Construct the objective function

$$[Maximize\ \alpha \sum_{i=1}^{N} B_i \times U_i - \beta \sum_{i=1}^{N} (T_i^{compute} + T_i^{transmit}) - \gamma \sum_{i=1}^{N} E_i], \tag{9}$$

where α、β and γ are the weight coefficients of each sub-objective, respectively.

C. *Constraints*

- Resource constraints: Set the maximum load for computing nodes and network links within each time period to avoid resource overload. Specifically, this includes restrictions on CPU, memory, storage, and network bandwidth resources. Among these, CPU utilization should not exceed 80%, memory utilization should not exceed 70%, storage utilization should not exceed 60%, and network bandwidth utilization should not exceed 90%.

- Task dependency constraints: Use a Directed Acyclic Graph (DAG) to represent task dependencies, ensuring that there are no cyclic dependencies between tasks.

- Latency constraints: Latency includes the execution time of tasks on computing nodes and the transmission time of data across the network. Set the maximum allowable latency 100ms.

- Task priority constraints: The lowest task priority is set to 1, and the highest task priority is set to 100. High-priority tasks should be allocated resources and scheduled before low-priority tasks.

- Load balancing constraints: Dynamically adjust task allocation based on response time, prioritizing shorter response times to balance the load across all computing nodes and network links.

D. Model Solving Approaches

For models of smaller scale that can guarantee linear properties, a Linear Programming (LP) solving approach is employed. Linear programming can find the optimal solution in polynomial time, and its solving process is relatively straightforward, making it suitable for certain subproblems in compute-network collaborative scheduling. In cases where there are nonlinear constraints or objective functions, Nonlinear Programming (NLP) methods are used, including the Interior-Point Method and Gradient Descent Method, to find local optimal solutions under nonlinear constraints and gradually approach a global optimum. For large-scale or high-dimensional optimization problems, heuristic algorithms and metaheuristic algorithms are utilized, including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). Heuristic algorithms simulate natural evolution and optimization processes to find near-optimal solutions in a relatively short time. For different stages and specific subproblems, hybrid solving methods are applied. For example, in the initial stage, heuristic algorithms are used to quickly find a feasible solution, followed by linear programming or nonlinear programming methods for fine-tuning optimization.

IV. ALGORITHM

Considering the dynamic change of the model objective function and the multi-objective property, we choose Mixed Integer Linear Programming (MILP) as the base algorithm. On this basis, we further introduce Genetic Algorithm (GA) as a supplement for dealing with complex search space and multi-objective optimization problems.

A. Algorithm

- Problem decomposition and modeling. Decompose the computing-network cooperative scheduling problem into several subproblems such as computational resource allocation, network resource allocation, and task scheduling, and establish corresponding mathematical models for each subproblem, which are described using the MILP framework.

- Initial Solution Generation and Adaptation Evaluation. Generate the initial population by randomly, each individual represents a scheduling solution, and use the MILP model to evaluate the fitness of the given individual and calculate its performance under the optimization objective.

- Selection and Crossover Operation. Based on the fitness value, good individuals are selected for crossover to generate new individuals. The crossover operation generates new individuals with different characteristics by exchanging part of the genes of the parent individuals to explore a better solution space[10].

- Variation operation and local search. In order to enhance the global search ability of the algorithm, a certain mutation probability is set to mutate the genes of some individuals[11]. At the same time, the simulated annealing local search strategy is introduced to fine-tune the mutated individuals and improve their fitness values.

- Termination conditions and iterative optimization of the solution. Set reasonable termination conditions, such as the maximum number of iterations or adaptation convergence index. When the termination condition is satisfied, output the current optimal solution as the final scheduling plan. Otherwise, continue the iterative optimization until the optimal solution is reached or the termination condition is satisfied.

B. Algorithm Steps

- Initialization Phase. Initialize the parameters of the algorithm based on the experimental environment and settings, including the initial population size, number of iterations, crossover probability, mutation probability, and so on.

- Initial Solution Generation. Based on the problem definition, randomly generate a certain number of initial solutions to form the initial population.

- Fitness Evaluation. Compute the objective function and calculate the fitness value for each individual in the population. The fitness value is determined by the objective function and measures the quality of each individual. Sort the population in descending order of fitness values, and preferentially retain individuals with high fitness.

- Selection Strategy. Use strategies such as roulette wheel selection and tournament selection to choose several individuals from the current population for crossover and mutation operations in the next generation.

- Crossover Operation. Perform crossover operations on the selected individuals based on a preset crossover probability to generate new individuals. Common crossover methods include single-point crossover and multi-point crossover. These new individuals are added to the next generation population through crossover operations.

- Mutation Operation. Conduct mutation operations on the newly generated individuals according to a preset mutation probability to increase the diversity of the population. Common mutation methods include bit mutation and swap mutation. The mutated individuals are added to the next generation population.

- Fitness Reevaluation. Recalculate the fitness value of each individual in the new population after crossover and mutation operations. Update the population based on fitness values, retaining individuals with high fitness and eliminating those with low fitness.

- Termination Condition Check. Check whether the algorithm has reached the preset number of iterations. Assess whether the fitness values of the population have converged to a certain range. If the termination conditions are met, output the optimal solution; otherwise, continue iterating.

- Output Results. Output the optimal solution after the iterations, including the best scheduling strategy and the corresponding objective function value.

C. *Algorithm 1 Hybrid Genetic Algorithm (HGA)*

```
1:  Input: Task set T , Compute node set C, Network node set N , Objective coefficients α,β,γ.
2:  Initialize: Population size P , Maximum iterations G, Crossover probability Pc, Mutation probability Pm.
3:  Randomly generate initial population S = {S1,S2,...,SP }.
4:  for generation = 1 to G do
5:      Step 1: Selection
6:      Select P parent solutions based on their fitness.
7:      Step 2: Crossover
8:      With probability Pc, perform crossover on selected pairs of parents to produce offspring.
9:      Step 3: Mutation
10:     With probability Pm, mutate selected offspring solutions.
11:     Step 4: Local Search using Simulated Annealing
12:     for each individual Si ∈ S (selected subset or all individuals) do
13:         Apply Simulated Annealing (Algorithm 2) to refine Si.
14:     end for
15:     Update the population with the refined solutions.
16: end for
17: Output: Optimal solution S∗ with the highest fitness.
```

D. *Algorithm 2 Simulated Annealing (SA)*

```
1:  Input: Solution S, Initial temperature Tinit, Cooling rate η, Maximum steps L.
2:  Initialize: T = Tinit, Sbest = S, Fbest = F (S).
3:  for step = 1 to L do
4:      Generate a neighbor solution Sneighbor from the current solution.
5:      Calculate the fitness Fneighbor.
6:      Compute ΔF = Fneighbor − F (S).
7:      if ΔF > 0 then
8:          Accept the neighbor solution: S ← Sneighbor.
9:      else
10:         Accept with probability exp(ΔF/T ).
11:     end if
12:     Update the best solution: Sbest = argmax(F (Sbest), F (S)).
13:     Reduce temperature: T ← η × T .
14: end for
15: Output: Optimized solution Sbest.
```

Algorithm Complexity. Assume that there are N computing nodes and M network nodes and the number of tasks is T. In the task allocation phase, each task needs to evaluate the resources of all computing nodes with a time complexity of O(T*N). Resource scheduling phase, it is assumed that each computing node needs to communicate with all network nodes with a time complexity of O(N*M). The load balancing phase, which requires traversing all tasks and computing nodes, has a time complexity of O(T*N). The total time complexity of the algorithm can be expressed as O(T*N + N*M + T*N), which simplifies to O(T*N + N*M) in the worst case. Assuming that the information size of each task, computational node and network node is C_t, C_n and C_m respectively, the total space complexity is $O(T*C_t + N*C_n + M*C_m)$. During the running of the algorithm, the space complexity of the intermediate results is O(T*N + N*M), assuming that the size of the task assignment matrix is T*N and the size of the matrix of the resource utilization is N*M. The total space complexity of the algorithm can be expressed as $O(T*C_t + N*C_n + M*C_m + T*N + N*M)$, which simplifies to O(T*N + N*M) in the worst case. In summary, when dealing with large-scale tasks and nodes, its time complexity and space complexity are more reasonable, and it is able to accomplish the computing-network coordination scheduling task under the premise of guaranteeing the system performance.

V. EXPERIMENTS

In terms of the experimental environment, this study was conducted on a server equipped with high-performance computing and network resources. The specific configurations are as follows: the CPU is an Intel Xeon E5-2680 v4 with a base frequency of 2.4 GHz and 14 physical cores; the memory is 128 GB DDR4; the storage device is a 1 TB NVMe SSD; and the operating system is Ubuntu 20.04 LTS. Additionally, the network environment used in the experiment was gigabit Ethernet, and Docker container management tools were deployed.

We selected the Google Cluster dataset[12] and the Alibaba Cluster dataset[13]. Each dataset underwent preprocessing, including steps such as noise data removal and normalization. The selection of experimental parameters was based on preliminary research and experience, including weight parameters for scheduling strategies, learning rates for optimization models, and the number of iterations. Each set of experiments was repeated multiple times, with the mean and standard deviation recorded. During the experiments, we focused on performance metrics such as algorithm convergence, computation time, and resource utilization. By comparing the experimental results under different parameter settings, we aimed to determine the optimal scheduling strategy and optimization model parameters. The specific parameters are as follows:

Number of Tasks. Between 5,000 to 10,000 task instances were selected from each dataset, covering task scheduling scenarios of various scales.

Task Types. Tasks were categorized into compute-intensive, data-intensive, and hybrid types.

Time Window. Various lengths of time windows were considered, ranging from 5 minutes (short-term) to 4 hours (long-term), to evaluate the model's scheduling effectiveness over different time spans.

Priority Settings. Task priorities were set according to different distribution methods, such as uniform distribution, normal distribution, and Poisson distribution, to simulate various scenarios in real applications.

We conducted multiple experiments with different datasets and parameter configurations to obtain stable and reliable results. Each experiment was repeated 10 times, and the average value was taken as the final result to minimize the impact of random factors on the experimental results.

A. *Optimized Model under Different Network Topologies*

We tested the performance of minimum cost routing scheduling strategy, real-time task scheduling strategy and the optimized model under different network topologies. The results are shown in Table 1. The optimized model outperforms the existing strategy in terms of system throughput.

TABLE I. NETWORK TOPOLOGY EXPERIMENT RESULTS

scheduling strategy	Small network (10 nodes, 15 links)		Medium network (50 nodes, 70 links)		Large network (100 nodes, 150 links)	
	Avg. latency (ms)	TPS (Mbps)	Avg. latency (ms)	TPS (Mbps)	Avg. latency (ms)	TPS (Mbps)
minimum cost routing	14.3	48.7	20.1	72.9	27.9	117.8
real-time task scheduling	12.5	50.2	18.7	75.4	25.3	120.5
optimization model	11.2	53.1	16.8	77.3	22.4	124.1

B. Optimized Model under Different Network Load Conditions

We tested the performance of minimum cost routing scheduling strategy, real-time task scheduling strategy and the optimized model under different network load conditions. The results are shown in Table 2. In most of the test scenarios, the optimized model exhibits low network transmission latency.

TABLE II. LOAD EXPERIMENT RESULTS

scheduling strategy	Light load (average traffic 20Mbps)		Medium load (average traffic 50 Mbps)		Heavy load (average traffic 100 Mbps)	
	Avg. latency (ms)	TPS (Mbps)	Avg. latency (ms)	TPS (Mbps)	Avg. latency (ms)	TPS (Mbps)
minimum cost routing	11.4	19.1	17.8	47.5	25.1	88.9
real-time task scheduling	10.2	19.5	16.5	48.3	23.7	90.2
optimization model	9.8	20.0	15.1	49.0	21.9	92.5

C. Optimized Model under Different Task Type Conditions

We tested the performance of minimum cost routing scheduling strategy, real-time task scheduling strategy and the optimized model under different task type conditions. The results are shown in Table 3. In most test scenarios, the optimized model table not only shortens the task completion time, but also effectively improves the task success rate.

TABLE III. TASK TYPE EXPERIMENT RESULTS

scheduling strategy	Compute-intensive task		Data-intensive task		Hybrid task	
	Time (s)	Success Rate (%)	Time (s)	Success Rate (%))	Time (s)	Success Rate (%)
minimum cost routing	37.5	90.8	45.3	87.9	40.9	88.6
real-time task scheduling	35.2	92.5	42.8	89.7	38.6	90.4
optimization model	33.1	95.0	40.5	91.3	36.3	93.2

Through comparative experimental analysis, traditional static scheduling strategies typically allocate resources and schedule tasks based on predefined rules and policies. Their advantages include simplicity of implementation and low computational overhead. However, they also have significant drawbacks, such as a lack of dynamism and adaptive capabilities, rendering them unable to cope with complex and volatile network environments. In contrast, the optimization model proposed in this paper demonstrates clear advantages in several areas. Firstly, in terms of dynamism and adaptability, the real-time data feedback mechanism enables rapid strategy adjustments, allowing it to handle complex and changing network environments effectively. Secondly, regarding global search capabilities, the improved heuristic algorithm effectively avoids local optima traps, enhancing optimization outcomes. Lastly, in terms of computational efficiency and resource utilization, the integration of machine learning's predictive capabilities with the efficiency of the optimization algorithm significantly reduces computational overhead and improves system performance. However, under extremely high-load conditions, the scheduling efficiency of the model decreases, resulting in some delay in task completion times.

VI. CONCLUSION

In this study, we propose a model to optimize the cooperative scheduling strategy of computing-network, and carry out in-depth discussion and experimental verification. By introducing an improved optimization algorithm, we effectively increase the resource utilization and task completion efficiency. In the process of model construction, constraints and objective functions under a variety of practical application scenarios are considered to ensure the wide applicability and accuracy of the model. The experimental results show that the optimization model performs well in a variety of environments, and compared with the traditional scheduling strategy, it not only significantly shortens the task completion time, but also achieves a better balance between the fairness and efficiency of resource allocation. However, the optimization model may consume a large amount of computational resources and time in the solution process when dealing with large-scale data and complex tasks[14]. Secondly, the actual effect of the optimization model depends largely on the accuracy of the input parameters and the assumptions of the model. Overall, this study has made some progress in the field of cooperative scheduling strategy optimization for computing-network, providing new ideas and methods for subsequent research. Future research can be devoted to constructing more comprehensive multi-objective optimization models and designing effective solution algorithms to achieve optimal multi-objective equilibrium under complex constraints.

REFERENCES

[1] Alibaba Cloud Computing Co. Ltd, CESI, (2018). White paper on technologies and standardization of edge cloud computing(2018). https://www.cesi.cn/images/editor/20181214/20181214115429307.pdf.

[2] Ethan C, Alicia W, Angie L, (2021). Edge Computing for Industrial AIoT Applications. https://cdn.madison.tech/wp-content/uploads/2021/03/02152822/moxa-edge-computing-for-industrial-aiot-white-paper-eng-2.pdf.

[3] Jin T J, Li W. (2022) An Intelligent Scheduling and Allocation Method of Big Data Computing Resources Based on Computing Power Network. Frontiers of Data and Computing, 6: 29-37.

[4] Zhong L J, Wang M. (2023) Blockchain-Enpowered Cooperative Resource Allocation Scheme for Computing First Network. Journal of Computer Research and Development, 60(4): 750-762.

[5] Wang Y J. (2011) Substation Planning Based on Initial Substation Site Decrease in Redundant Meshes. Chongqing University, 1:68.

[6] Duan X D, Yao H J, Fu Y X, Lu L, Sun T. (2021) Computing force network technologies for computing and network integration evolution. Telecommunications Science, 37(10): 76-85.

[7] Li, X. (2011) A Study on Delay Compensation Control Strategies for Networked Control Systems Based on Predictive Control. Jinan University, 11:80.

[8] Li M X, Cao C, Tang X Y, He T, Li J F, Liu Q Y. (2020) Research on Edge Resource Scheduling Solution for Computing Power Network. Frontiers of Data and Computing, 2(4): 80-91.

[9] Liu Z N, Li K, Wu L T, Wang Z, Yang Y. (2020) Cost Aware Task Scheduling in Multi-Tier Computing Networks. Journal of Computer Research and Development ,57(9): 1810-1822.

[10] Zeng S Y. (2022) Study on Joint Optimization of Multi-capable Workers and Task Allocation in Seru Production System under Fairness Perspective. Jiangsu University of Science and Technology, 2:119.

[11] Cao, W H. (2019) Research on Optimization of Operation Organization of Railway Container Central Station Based on Customer Classification. Beijing Jiaotong University, 1:135.

[12] Muhammad Tirmazi, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steve Hand and Adam Barker. (2019) Google Cluster Dataset. https://github.com/google/cluster-data.

[13] Alibaba Group. (2018) Alibaba Cluster Dataset. https://github.com/alibaba/clusterdata/tree/v2018/cluster-trace-v2018.

[14] Bian L H, Li D Y, Chang X Y, Suo J L.(2023) Theory and method of large-scale computational reconstruction. Laser & Optoelectronics Progress,60(02):11-28.