Journal of Network Intelligence ©2026 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **11**, Number **1**, February **2026**

A Cross-Source Scheduling Method for Computer Heterogeneous Big Data Based on Improved Heuristic Algorithm

Min-Juan Liu[1,*], Xi-Zhi Zhang[1]

[1]College of Information Engineering,
Zhengzhou Shengda University, Zhengzhou 451191, P. R. China
{25741190, 1908879334}@qq.com

He-Ling Cao[2]

[2]College of Information Science and Engineering,
Henan University of Technology, Zhengzhou 450001, P. R. China
caohl@haut.edu.cn

Kuang Jing[3]

[3]Bulacan State University, Malolos 3000, Philippines
704161029@qq.com

*Corresponding author: Min-Juan Liu

Received December 11, 2024, revised March 19, 2025, accepted June 22, 2025.

Abstract. *Rapid development of computer technology has progressively made heterogeneous big data and cross-source scheduling issues the most important obstacles large-scale computing systems must solve. The basis for the cross-source scheduling solution presented in this work is an improved heuristic algorithm sometimes known as a genetic algorithm (or GA for short). This approach aims to efficiently manage the job scheduling issue inside the framework of a heterogeneous resource environment. In this study, an adaptive crossover and mutation technique is proposed with the aim of solving the difficulty of the standard genetic algorithm addressing the "local optimal solution". This method allows one to strike a compromise between the exploration and exploitation powers of the algorithm by means of continuous change of the crossover and mutation rates. The results of the studies show that, in terms of the efficiency with which jobs are scheduled, the distribution of load, and the use of resources, the suggested AD-CSA model shows noteworthy benefits over the conventional scheduling method. Furthermore proof of their existence is the rather great nature of these advantages. This study also looks at the model's application possibilities in settings with heterogeneous massive data volumes. This results in fresh ideas and possible solutions for the topic of computer science dealing with cross-region scheduling.*

Keywords: heterogeneous big data; cross-source scheduling; GA; heuristic algorithm.

---

1. Introduction.

As information technology has quickly expanded data, particularly in heterogeneous big data environments, adequate storage, analysis, and scheduling of data has become a major issue [1]. Large-scale distributed systems are finding favour in cloud computing, edge computing, and Internet of Things. In these systems, scheduling diverse computer resources is challenging [2]. Popular subjects in academics and business are how to make use of several resources to boost job execution speed, scheduling efficiency, and resource utilisation.

1.1. **Related work.**

Main research subjects have been optimizing job scheduling, computational efficiency, and resource use. Load-balancing scheduling techniques for heterogeneous large data environments have been created many times to dynamically change computing task allocation to lower task waiting time and resource idle time [3], thereby improving system performance. Some researchers have presented a multi-level scheduling system that maximizes compute resource allocation and data flow across several storage nodes and network channels in big data settings [4], so enhancing heterogeneous big data platform processing capabilities and stability [5].

To effectively carry out operations in various environments in the cross-source scheduling problem, researchers investigate ways to coordinate platforms, data sources, and computing resources [6]. Data flow graph-based scheduling systems that dynamically change task execution order depending on dependencies have been proposed by several scholars [7]. These techniques improve system performance, lower cross-source scheduling lag, and guarantee task execution reasonableness. Cloud computing cross-origin scheduling systems, for example, use real-time resource monitoring and load prediction to change task scheduling order, so improving data access and computation efficiency.

Due to their efficiency and adaptability, which enhances large-scale task scheduling performance, heuristic algorithms are widely used in heterogeneous computing resource task scheduling challenges. Task scheduling optimization finds use for Genetic Algorithm (GA) [8], Ant Colony Algorithm (ACO) [9], and Particle Swarm Optimisation (PSO) among other things [10]. To determine the global optimal solution, GA runs natural selection and genetic variation simulations. Its crossover and mutation techniques enable it to traverse the search space and leap out of the local optimal solution, therefore addressing the computational cost and slow solution speed of standard scheduling techniques.

Many researchers have devised GA enhancing strategies to overcome its local optimal solutions and limited convergence time by use of heuristic algorithms. An adaptable technique to change GA's crossover and mutation rates helps it to balance exploration and exploitation at several phases [11], so enhancing its capacity to address large-scale job scheduling challenges in heterogeneous computing settings. Crucially for addressing scheduling restrictions in large-scale distributed computing, these works provide creative solutions for heterogeneous big data and cross-source scheduling problems.

1.2. **Contribution.**

The following are the key ideas suggested in this work regarding the present computer job scheduling issues:

(1) Improved GA. This paper presents an adaptive crossover and mutation strategy to dynamically change crossover and mutation rates depending on generation fitness distributions, hence optimizing GA performance. Overcoming heterogeneous resource scheduling's unequal load and high latency problems, the approach balances exploration and development, improves algorithm convergence speed and global search.

(2) Cross-source scheduling method proposal. This work presents a cross-source scheduling system that effectively arranges tasks among several computing resources. By means of resource allocation and scheduling optimization, this approach manages data and tasks from several sources, overcomes low resource utilization and long task execution times in large-scale heterogeneous data environments.

(3) The introduction of comprehensive evaluation metrics. The main evaluation indicators in this paper's whole scheduling technique are load balance, efficiency of scheduling, and resource use. These metrics provide a scientific basis for algorithm modification and

optimisation by quantifying the running efficiency of the algorithm and the reasonable use of computer resources in the scheduling outcomes.

2. Theoretical analysis.

2.1. Heterogeneous big data.

2.1. Heterogeneous big data. Heterogeneous big data is the complexity of data sources coming from several kinds of data sources [12], which may have distinct storage structures, formats, access modes and data types. Among the several data sources could be structured data (such as relational databases), semi-structured data (such as XML and JSON files), and unstructured data (text documents, photos, and videos). Extensive information extraction, support of decision-making, and maximum business process optimization depend on efficient administration and scheduling of these heterogeneous data. In the computer sector, particularly in remote computing and data scheduling systems, how to efficiently handle this heterogeneous data to meet effective data transmission [13], storage and querying needs is the key to achieving cross-source scheduling, see Figure 1.

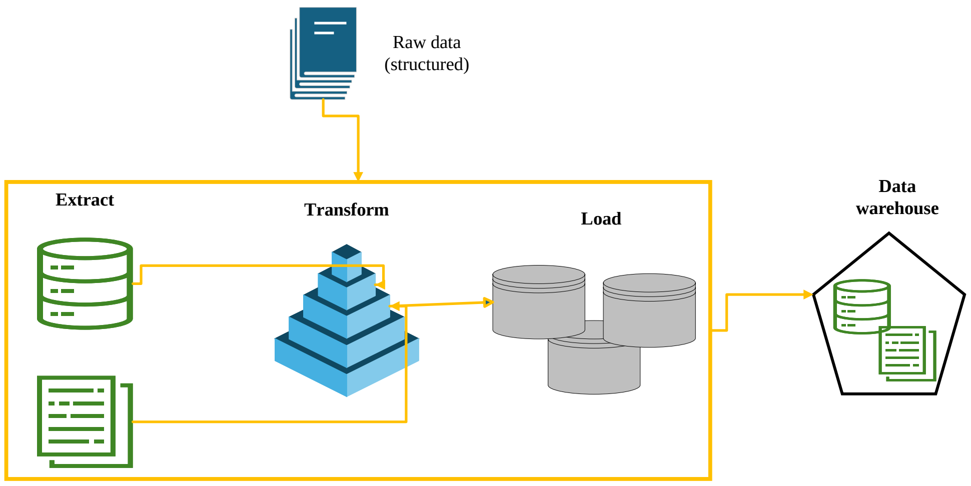

Figure 1. Heterogeneous big data processing process

First consideration in the processing of heterogeneous big data should be the time consumption of gathering data from several data sources [14]. Given the quantity of data and network bandwidth, let the time for task t_i to acquire data from data source D_k be t_{ik}, which is expressed as Eq:

$$t_{ik} = \frac{|D_k|}{B_{ik}} \tag{1}$$

where $|D_k|$ is the data from data source D_k and B_{ik} is the bandwidth upon task of accessing data from data source t_i. Choosing the data source and transmission method properly in task scheduling is essential since the network capacity limitation may lead the data transmission time to become a system performance bottleneck.

Second, another crucial problem is the way diverse data is integrated [15]. Tasks have to combine data from many sources since data sources and formats differ. Let task t_i acquire data from K data sources $D_1, D_2, \ldots, D_K$; its integrated data volume D_{new} can be stated as:

$$D_{new} = \sum_{k=1}^{K} w_k \cdot |D_k| \tag{2}$$

where w_k represents the weight of the data source D_k, therefore indicating its quality or priority. The method shows how to combine and weigh different data sources to maximize data use efficiency. Under this procedure, the task scheduling algorithm must choose the data sources, fairly distribute the data transfer, and prevent the creation of duplicated data.

Task execution performance in heterogeneous big data processing is strongly influenced by query latency t_{query}^{ij}. It can be stated assuming that the data amount, bandwidth, and network latency define the query latency of task t_i when querying data source D_j as follows:

$$t_{query}^{ij} = \frac{|D_j|}{B_{ij}} + \delta_{ij} \tag{3}$$

where B_{ij} is the bandwidth; δ_{ij} is the network delay between the data source D_j and the task t_i; $|D_j|$ is the data source size. This formula captures the significance of query performance in cross-source scheduling, particularly in cases of high volume of heterogeneous data when optimising query latency is essential to raise the general system efficiency.

Furthermore considering the resource allocation issue of computer nodes, load balancing is another difficulty to get effective processing of heterogeneous large data. On a design computing node R_j, the load utilisation rate ρ_j is the ratio of the total job time to the computational capacity, eq:

$$\rho_j = \frac{\sum_{i=1}^{N} E_{ij} \cdot x_{ij}}{Capacity(R_j)} \tag{4}$$

where $Capacity(R_j)$ is the computing capacity of node R_j; E_{ij} indicates the execution time of job t_i on computing node R_j; x_{ij} is an indication variable of whether task t_i is allocated to node R_j. Through efficient load balancing, some nodes can be essentially avoided from being idle or overloaded, therefore enhancing the general system scheduling efficiency.

At last, one cannot overlook the storage limits of heterogeneous large data [16]. The storage capacity of a computing node directly influences the data processing in a distributed environment. Let the storage capacity constraint of task t_i on computing node R_j be S_j; then, the total data volume of the task after getting data from several data sources should not exceed the storage capacity of the node. One could represent the limitation as:

$$\sum_{k=1}^{K} x_{ik} \cdot |D_k| \leq S_j \tag{5}$$

x_{ik} shows if the work t_i reaches the data source D_k, where S_j is the storage capacity of the computational node R_j. Reasonable storage limits guarantee that, while running tasks, the computing node does not surpass its capacity for storing, therefore preventing system performance reduction.

2.2. **Cross-source scheduling.** Heterogeneous big data systems need cross-origin scheduling to maximize task execution efficiency and resource use given data storage on several sources and computing nodes. Cross-source scheduling improves system computational efficiency by coordinating activities and data across several data sources and processing nodes, hence lowering task execution time, data transmission latency, and resource waste [17].

The scheduling of tasks not only addresses data transfer across several nodes and sources but also involves thorough computational resource allocation considering the unequal distribution of data and computational resources in heterogeneous big data systems. One may represent the scheduling time of a task as:

$$t_{\text{task}}^{ij} = t_{\text{trans}}^{ijk} + t_{\text{exec}}^{ij} \tag{6}$$

From data source D_k, t_{trans}^{ijk} is the transmission time of task t_i; t_{exec}^{ij} is the execution time of job t_i on computation of node R_j. Cross-source scheduling not only maximizes the transmission time t_{trans}^{ijk} but also has to take calculation time t_{exec}^{ij} on the computation node R_j into account, therefore determining the total job completion time.

Task scheduling depends on the choice of data sources and computation nodes being quite important [18]. For every job t_i, we wish to choose the best data source D_k to shorten the data transmission time, and distribute the computational resources suitably in line with the load of the computational nodes. The cross-source scheduling issue can be stated as: by optimising the cost of data source selection:

$$\text{Minimize} = \sum_{i=1}^{N} \sum_{k=1}^{M} C_k \cdot x_{ik} \tag{7}$$

where x_{ik} is an indicator variable denoting if task t_i reads data from data source D_k; C_k is the transmission cost of data source D_k. The scheduling algorithm can choose the best data source by varying the value of x_{ik}, therefore lowering the general scheduling cost.

Cross-source scheduling depends also much on load balancing of computational nodes. The execution time of the task can be stated assuming that the processing capability of every computing node R_j is P_j and the computational complexity of task t_i is C_i as:

$$t_{\text{exec}}^{ij} = \frac{C_i}{P_j} \cdot (1 + \gamma_{ij}) \tag{8}$$

Reflecting the load of the computer node P_j, γ_{ij} is the delay factor when the task is carried out there. Reasonable load balancing helps to minimize the overloading or idling of particular nodes thereby guaranteeing the effective running of the system.

Cross-source scheduling also has to meet the limitation on storage capacity. We derive the following constraint to prevent storage overload assuming that the storage capacity of every computing node R_j is S_j and task t_i needs to occupy storage space S_i:

$$\sum_{i=1}^{N} x_{ij} \cdot S_i \leq S_j, \forall j \tag{9}$$

The formula guarantees that the storage need of every compute node R_j does not beyond its capacity S_j, therefore preventing data overflow and waste of storage resources.

At last, the dependencies among jobs present even another difficulty in cross-source scheduling. By considering the dependencies and priority of activities, we can guarantee that chores are carried out in the right sequence:

$$S_i \geq \max_{j \in \text{Pred}(i)} F_j + \pi_i \tag{10}$$

where $\text{Pred}(i)$ is the collection of predecessor tasks for job t_i; F_j is the completion time of predecessor task t_j; π_i is the task's priority. This formula guarantees that execution conflicts are prevented and inter-task dependencies are satisfied.

Therefore, an approach able to manage the complexity and dynamic and make optimal resource allocation decisions across several data sources and compute nodes is necessary to enable effective cross-source scheduling.

2.3. **Heuristic algorithms.**

In the parts before we covered heterogeneous large data and cross-source scheduling. Especially GA, improved heuristic algorithms help to effectively optimize job scheduling [19]. Ideal for heterogeneous computing resources and cross-origin scheduling challenges, GA rapidly identifies near-optimal solutions in large-scale solution spaces by mimicking natural selection and inheritance (Figure 2).

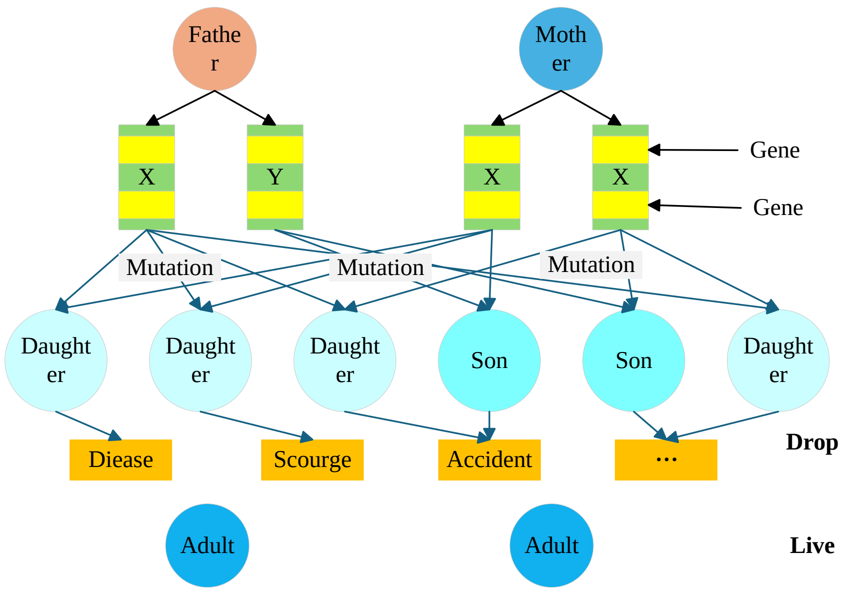

Figure 2. Example of genetic algorithm model

In classic GA, individual representation is absolutely important [20]. Assume that any scheduling method may be expressed as a chromosome C, which codes an N-length gene string:

$$C = (c_1, c_2, \dots, c_N) \tag{11}$$

where c_i stands for the resource or schedule guide to which i have been allocated. Selection, crossover and mutation constitute the main GA processes. Initially, the selection process chooses the parent individual depending on the fitness value $f(C)$:

$$f(C) = \sum_{i=1}^{N} w_i \cdot T_i \tag{12}$$

where T_i is the execution time of job i and w_i is its weight. The GA can bring the search process toward the best solution by choosing parents who have better fitness.

The crossover operation creates new humans by combining two parent individuals' genomes [21]. The resultant child individuals C_3 and C_4 can be stated assuming that the parent individuals C_1 and C_2 cross at position k as:

$$C_3 = (c_1, c_2, \dots, c_k, c_{k+1}, \dots, c_N) \tag{13}$$

$$C_4 = (c_1, c_2, \dots, c_{k-1}, c_k, \dots, c_N) \tag{14}$$

Good traits from the parent to the offspring individuals may be transferred thanks to crossover operations [22]. Conversely, the mutation procedure generates fresh solutions by randomly altering the values at certain places in the gene. Mutation operation follows a formula:

$$C' = (c_1, c_2, \dots, c_i', \dots, c_N) \tag{15}$$

where c_i' represents the gene c_i mutation value. Appropriate variation helps GA to avoid local optimal solutions and thereby increase the capacity of the global search.

In the cross-source scheduling problem, GA not only maximizes the scheduling order of jobs but also considers the delay of data transmission and the load balancing of computational resources. The transmission delay can be computed using assuming that the transmission time of task t_i from the source node S_i to the target node R_j is t_{trans}^{ij}:

$$t_{\text{trans}}^{ij} = \frac{D_i}{B_{ij}} + \alpha \cdot L_j \tag{16}$$

where D_i is the task data size; B_{ij} is the bandwidth between the target and source nodes; L_j is the load of the target node; α is the coefficient of the transmission delay. By means of optimal task scheduling and resource allocation, GA aims to minimise the total completion time T_{total} of the task so attaining the improvement of the scheduling efficiency.

Adaptively changing crossover and mutation rates based on the fitness distribution of every generation helps the GA to dynamically balance exploration and exploitation during the search phase. With this approach, GA can more quickly and precisely tackle heterogeneous large data and cross-source scheduling problems.

We thus offer enhanced adaptive crossover and mutation techniques that dynamically change the crossover and mutation rates depending on the fitness distribution of every generation in order to increase the performance of the GA even more. This enhancement increases the global search capacity and convergence speed of the method as well as helps the GA to more dynamically balance the exploration and development process.

3. A cross-source scheduling model for computer heterogeneous big data based on improved heuristic algorithms: AD-CSA.

3.1. Model framework.

The AD-CSA modeling framework consists in population initiation, selection, crossover, mutation, adaptive tuning methods, and termination criteria. Every division takes care of a certain chore, guaranteeing precision and effectiveness of schedule.

(1) Population initialisation. The phase of population initialisation generates an initial population at random by means of work scheduling plans [23]. Every person stands for a task scheduling system; the way the person is coded depends on the mapping relationship between tasks and resources. P_0 is the starting population size; individual genetic coding reflects assignment of tasks and resources and task scheduling order.

(2) Selection operation. By assessing their fitness, the selection process chooses which people belong in the next generation. The fitness function F is defined as the merit of the scheduling scheme, where the merit consists in the task completion time (Makespan) and the degree of resource load balance. One may represent the fitness function as follows:

$$F = \frac{1}{\text{Makespan} + \alpha \cdot \text{Load imbalance}} \tag{17}$$

where α is a control factor for load imbalance effect on fitness. Roulette selection or tournament selection process helped the parent individuals to be chosen depending on the fitness worth.

(3) Crossover operation. By mixing the genes of two parent people, crossover operation is utilized in GA to investigate the solution space by producing new offspring individuals [24]. This work presents a crossover method based on fitness weighting to improve the searching capability of the algorithm. More precisely, the person with better fitness is chosen as the parent to raise the likelihood of the superior genes being passed on to the children. With two parent individuals x_1 and x_2 assuming that the offspring person x_3 is produced by crossover operation, the crossover process can be stated by the following equation:

$$x_3 = x_1 \oplus (\text{Rand}(0,1) \cdot (x_2 - x_1)) \tag{18}$$

where $\oplus$ represents the crossover operation and $\text{Rand}(0,1)$ is a random number evenly spaced between 0 and 1, which is dynamically changed the fraction of crossover. This crossover increases the quality of the answer by means of information exchange amongst many people while preserving the stability of the superior genes.

(4) Mutation operation. By randomly altering some gene values of individuals, mutation operation seeks to boost population variety and avoid the algorithm from descending into a local optimal solution [25]. This work suggests an adaptive mutation technique to improve the exploring capacity of mutation operation even further. The fitness distribution of the population in every generation determines dynamically the mutation rate. Whereas the mutation rate is lowered for precise search when the fitness rises quickly, it is raised for more general search when the fitness varies slowly. Should individual x_i need to be changed throughout the mutation process, its update formula is:

$$x_i' = x_i + \delta(\text{Rand}(-1, 1)) \tag{19}$$

The direction and extent of the mutation are ascertained using $\text{Rand}(-1, 1)$, a random number between -1 and 1, and δ, the mutation step size. The approach can dynamically modify the variance's strength to fit several search phases and maximize the algorithm's performance.

(5) Adaptive crossover rate and variation rate strategy. This work presents an adaptive crossover rate and mutation rate adjustment mechanism to improve the search efficiency of GA in complex heterogeneous big data cross-source scheduling even further. The method dynamically adjusts the likelihood of crossover and mutation operations by analyzing the fitness distribution of the population in every generation, thereby better balancing exploration and exploitation. In particular, the variations in crossover and mutation rates are tuned in line with changes in individual fitness, therefore enabling the algorithm to prevent stagnation close to the local optimal solution and enhance the worldwide search capacity.

Dynamic adjustment of the crossover rate P_{cross} and the variation rate P_{mut} follows changes in the fitness. The crossover rate and mutation rate are computed using this method assuming that the population's fitness distribution satisfies specific criteria at generation t:

$$P_{\text{cross}}(t) = \frac{1}{1 + \exp(-\gamma \cdot (F_{\text{avg}}(t) - F_{\text{target}}))} \tag{20}$$

$$P_{\text{mut}}(t) = \frac{1}{1 + \exp(-\delta \cdot (F_{\text{avg}}(t) - F_{\text{target}}))} \tag{21}$$

where $F_{\text{avg}}(t)$ is the average fitness of the current generation, $F_{\text{target}}(t)$ is the target fitness, and γ is the regulation parameter.

(6) Termination conditions and convergence analysis. We propose a dynamic termination condition to guarantee that the GA pauses at the proper moment and prevent needless computation. Usually setting a predetermined number of iterations or fitness threshold as the termination condition, traditional GA may not be able to meet the requirements of various issues. This work presents a termination condition based on convergence analysis to increase the adaptability and flexibility of the method.

The degree of fitness change of the present population determines the environment of the termination condition. The algorithm can be regarded to have converged and the iteration is terminated when the change in fitness across several successive generations is less than a specified threshold. Particularly, the rate of change in fitness ΔF is defined as the change in optimal fitness between two generations according to the following formula:

$$\Delta F = F_{\text{best}}(t) - F_{\text{best}}(t - 1) \tag{22}$$

where $F_{\text{best}}(t)$ is the fitness value of the ideal individual in the t^{th} generation. The method is regarded as convergent and can be stopped when ΔF is smaller than the specified threshold ϵ.

Furthermore able can be stopped depending on $T_{\max}$, the maximum number of iterations, or on obtaining a predetermined fitness value, F_{target}. The algorithm pauses iteratively and generates the current optimal work scheduling plan when one of these criteria is satisfied. By means of such adaptive termination criteria, the method can eliminate pointless repetitive computations and guarantee high efficiency, so enhancing computational economy.

3.2. **Assessment indicators.** This work aims to assess the performance of heterogeneous big data cross-source scheduling techniques based on enhanced heuristic algorithms using selected important evaluation measures.

(1) Scheduling efficiency. In a heterogeneous context, scheduling efficiency finds the equilibrium between the usage of resources and the speed at which a scheduling algorithm finishes a task. It shows how well the method manages big chores. Greater scheduling efficiency suggests that the method can reasonably arrange computational resources and finish the work more fast. The calculation is:

$$E_{\text{sched}} = \frac{\sum_{i=1}^{N} T_{\text{task}}(i)}{T_{\text{total}}} \tag{23}$$

where $T_{\text{task}}(i)$ is the time needed for task i to finish; T_{total} is the overall scheduling time; N is the overall number of tasks.

(2) Load Balance. An indication of whether computing resources are distributed fairly throughout task scheduling is load balancing degree. Load balancing helps to lower the idle condition of some computational nodes and the overload of some other nodes in heterogeneous large data settings, therefore enhancing the general system performance. The pattern is:

$$L_{\text{balance}} = \frac{\sum_{i=1}^{M} |T_{\text{load}}(i) - \bar{T}_{\text{load}}|}{M \cdot \bar{T}_{\text{load}}} \tag{24}$$

where $\bar{T}_{\text{load}}$ is the average load of all the nodes; $T_{\text{load}}(i)$ is the load of node i; M is the number of nodes.

(3) Resource utilisation. The degree of efficient utilization of every computational resource in the task scheduling process in a heterogeneous system is gauged by means of resource utilisation. High resource utilization helps the scheduling system to effectively use the system resources and prevent waste or idleness of resources. The calculation is:

$$R_{\text{util}} = \frac{\sum_{i=1}^{M} T_{\text{used}}(i)}{\sum_{i=1}^{M} T_{\text{available}}(i)} \tag{25}$$

where $T_{\text{used}}(i)$ is the time spent by node i; $T_{\text{available}}(i)$ is the time node i has at hand.

By means of these extensive evaluation criteria, we can evaluate and assess the performance of several scheduling techniques holistically, therefore provide a basis for choosing the optimal algorithm. Using these evaluation criteria not only clarifies the algorithm strengths and shortcomings but also offers a consistent evaluation structure for next studies.

4. **Performance testing and analysis.**

4.1. **Experimental environment.** Run on a clustered computing environment with heterogeneous Intel Xeon E5-2670 v3 CPUs, 64 GB RAM, and 2 TB SSD storage, this experiment runs with every node in order to maximize data movement and computing resource sharing is 1 Gbps Ethernet linked. Running Python 3.8 using NumPy, SciPy, and custom-implemented extended GA modules for computation and data processing. Ubuntu 20.04 LTS through parallelizing scheduling techniques on numerous nodes, the tests replicate heterogeneous computing.

4.2. **Experimental dataset.** Task and resource constitute the two components of the dataset used in this experiment. The assigned work consists in several jobs with different computational demand, data transfer quantity, and priority level. On the CPU clock cycles required to complete the task on the compute node is computational demand. Task execution calls for data flow spanning bytes between nodes. Priority denotes urgency, hence high-priority tasks should be finished first. Every job also has an execution time. The task set may consist of hundreds to thousands of activities depending on the experimental design to test the algorithm in mass task scheduling.

Every computing node in the resource set boasts different storage capacity and processing capability. Computational power and storage capacity of a node depend on CPU frequency, RAM size, and disk space. The resource set comprises several nodes with diverse computing capability, some of which can manage high-computing-density tasks and others low-computing-density workloads, so replicating the heterogeneous environment in real-world application scenarios. The resource set comprises 10 to tens of computing nodes, depending on experimental needs.

Table 1. Example task set data

Task ID	Computational Demand (CPU Cycles)	Data Transfer Volume (Bytes)	Priority	Execution Time (Seconds)
T1	5000	20000	High	10
T2	2000	5000	Medium	5
T3	8000	10000	Low	15
T4	1500	8000	High	4
T5	10000	25000	Medium	20

Table 2. Example resource set data

Resource ID	Processing Power (CPU Frequency)	Storage Capacity (GB)	Network Bandwidth (Mbps)
R1	3.0 GHz	64	1000
R2	2.4 GHz	128	1000
R3	3.2 GHz	32	500
R4	2.5 GHz	64	1000
R5	2.8 GHz	128	500

4.3. **Experimental process and results.** Based on enhanced heuristic algorithms, we create a series of tests in the experimental phase of this work to assess and analyze the performance of the proposed computer heterogeneous big data cross-source scheduling approach (AD-CSA). These tests seek to fully confirm in practical settings the scalability, efficiency, and efficacy of the AD-CSA paradigm. Three well-crafted tests, each aiming at a certain research question and performance criterion, help us to reach this aim.

(1) Task scheduling effectiveness comparison experiment. This experiment aims to confirm the task scheduling optimisation efficiency of AD-CSA in a heterogeneous computing resource environment by means of comparison with the conventional GA and simulated annealing algorithm, so assessing the benefits of AD-CSA in resolving resource load inequality and scheduling delay.

In the trials, we design several work sets on a heterogeneous computing resource environment. Whereas resource sets have compute nodes with varying computational and storage capacity, task sets have tasks of different sizes and complexity. Task sets are planned using AD-CSA, standard GA, and simulated annealing algorithm (SA), and execution time, resource use, and load balancing are documented for every method to guarantee correctness in several rounds of trials. The scheduling performance of every algorithm reveals the advantages of AD-CSA in job scheduling, most especially in terms of task execution time, resource use, and scheduling efficiency.

Figure 3 contrasts simulated annealing task scheduling performance for varying job sizes with AD-CSA, ordinary GA. We assessed our investigations using measures of resource utilization, load balance, and scheduling efficiency.

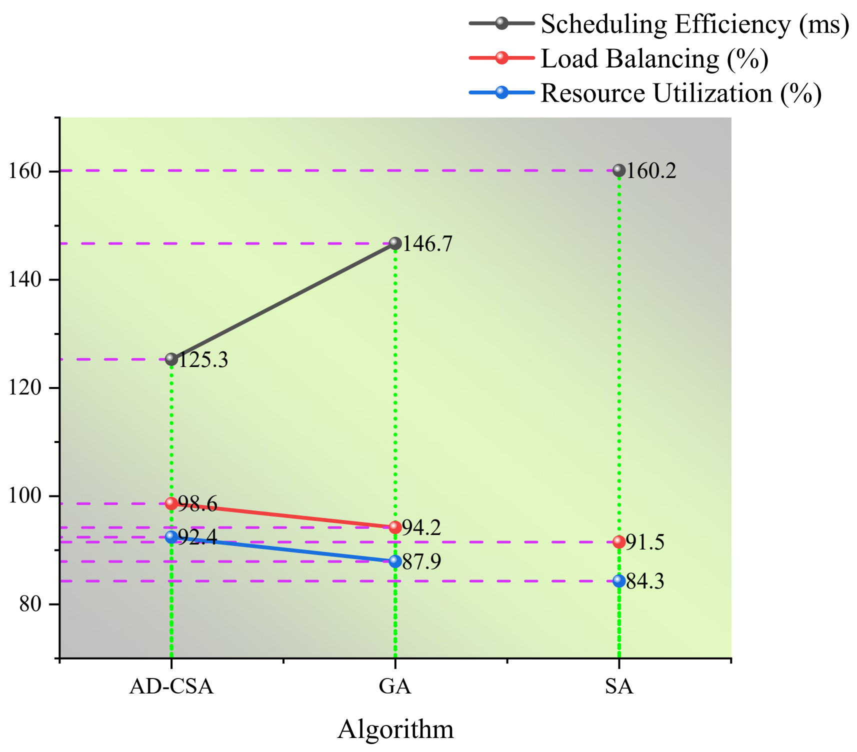

Figure 3. Experimental results of task scheduling effect comparison

In terms of scheduling efficiency, load balancing degree, and resource use, the experimental findings clearly show that AD-CSA beats both the normal GA and the simulated annealing technique. Particularly, AD-CSA increases the scheduling efficiency by roughly 14.6% (in comparison with normal GA) and roughly 21.8% (in comparison with simulated annealing.). Load balancing-wise, AD-CSA scores 98.6%, well over 94.2% for conventional GA and 91.5% for simulated annealing. Furthermore superior than the other two methods is AD-CSA's 92.4% in terms of resource economy.

(2) Algorithm convergence experiment. The aim of this work is to assess the convergence speed and accuracy of AD-CSA and GA in the job scheduling problem.

By tracking the change in its fitness under various iteration counts, the experiment tests whether each method can identify the near-optimal solution inside a constrained number of iteration cycles. Running AD-CSA and GA correspondingly, the experiments follow the identical starting circumstances and parameter settings; until the maximum number of iterations reaches 1000, the fitness values are recorded for every 100 iterations. All experiments were carried out thirty times separately to guarantee the consistency of the experimental results; the average value was at last taken for comparison. Figure 4 displays the experimental outcomes.

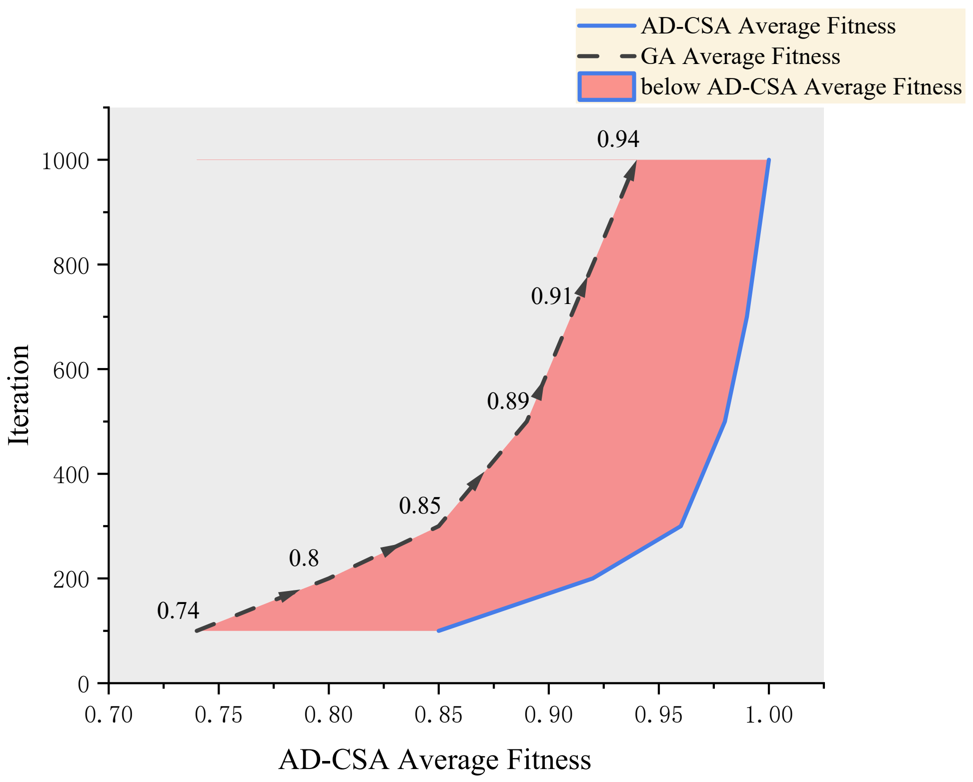

Figure 4. Experimental results of algorithm convergence

AD-CSA converges better than GA according to experimental data. After 100 iterations, AD-CSA's fitness value approaches 0.85 after which it approaches the optimal solution and 1.00 after 1000 iterations, therefore showing faster convergence. After 1000 cycles, GA's fitness value stays 0.94—less than the ideal response. AD-CSA can converge faster than GA, better schedule tasks, and identify a better answer in less iterations.

(3) Cross-Source Scheduling Performance Experiment. This experiment is to assess the efficacy of AD-CSA in handling cross-source scheduling challenges mostly by contrasting the scheduling effects of AD-CSA and conventional GA in heterogeneous big data environments.

In order to assess cross-origin task scheduling efficiency, the experiment runs task scheduling across sources with varying job loads and resource constraints. Under different job size, trials investigate the scheduling time, resource use, and load balancing degree of the two strategies. To investigate AD-CSA's flexibility in cross-source scheduling, the trials included different inter-source communication delays. Under 30 separate trials, the tests capture scheduling time, load balancing degree, and resource use; they then evaluate the two algorithms under several conditions. Figure 5 displays experimental findings.

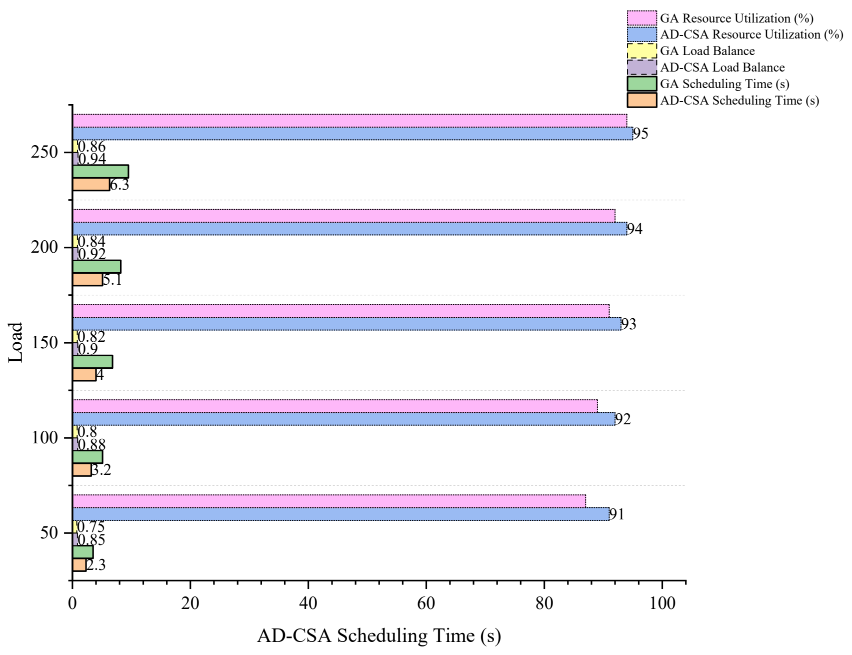

Figure 5. Cross-source scheduling performance experiment results

Experimental results reveal that AD-CSA greatly beats GA in cross-origin scheduling; when the task load and resource demand rise, AD-CSA's scheduling time is far lower than that of GA and it can more effectively complete the scheduling assignment. AD-CSA likewise excels in terms of resource economy and load balance. Specifically, AD-CSA's load balance and resource utilisation are higher than GA's in all experimental configurations, indicating that AD-CSA can better cope with the cross-origin scheduling issue in heterogeneous big data environments, optimise task allocation, and so improve the general resource utilisation efficiency. Particularly in cases with a heavy work load, the difference between GA and AD-CSA in terms of scheduling time, load balancing and resource utilisation is more clear. This suggests that AD-CSA can offer a more effective scheduling solution and is more flexible in handling significant cross-origin scheduling.

Three studies on the AD-CSA model find that it shines in diverse big data contexts for cross-origin scheduling. Especially as task size grows, AD-CSA performs well in load balancing, resource utilisation, and scheduling time in the scheduling efficiency experiment; it also prevents premature convergence in the algorithm convergence experiment. AD-CSA performs well in load balancing, resource utilisation, scheduling time and optimising resource allocation in cross-origin scheduling performance testing. Practically, the AD-CSA method solves heterogeneous computing resources and large-scale cross-origin scheduling problems.

5. **Conclusion.** In this work, we construct a model called AD-CSA and present a cross-origin scheduling technique for heterogeneous big data in computers depending on enhanced heuristic algorithms. The findings of a sequence of experimental validation of the model reveal that AD-CSA significantly solves the cross-origin scheduling issue in heterogeneous big data environment.

Still, this study has several restrictions. First of all, the model performs better when the size of the scheduling assignment is small but may have a performance bottleneck handling very large-scale projects. Second, AD-CSA has a high computational complexity, which must be further optimized to fit various application needs upon actual implementation.

Future studies might be conducted in the following lines:

(1) Introduction and optimisation of hybrid algorithms. Although GA cannot solve big-scale complicated problems, our work bases on enhanced GA as the scheduling method. Future study should combine PSO with SA to create a hybrid optimization technique. Combining several techniques increases efficiency and accuracy of scheduling. Combining GA's local search with PSO's global search will help to enhance task scheduling in heterogeneous big data environments.

(2) Combination of deep learning and scheduling algorithm. Future research can combine deep learning algorithms with scheduling systems as deep learning technologies develop to enhance scheduling intelligence. Deep neural networks (DNN), convolutional neural networks (CNN), and other methods extract task features and computational resource state information in order to maximize scheduling.

(3) Research on large-scale datasets and real-time scheduling problems. This work uses a quite modest dataset for trials; future work can assess scheduling algorithms in bigger and more sophisticated heterogeneous data contexts. Specifically, when dealing with real-time task scheduling, one must get the execution state of the job and the system load in real time and investigate how to develop accurate and efficient scheduling algorithms in highly dynamic changing contexts and massive data streams.

Looking ahead the future and believing that with our combined knowledge and efforts we can create new research frontiers and unleash new potential for big data and cross-source scheduling challenges as we keep stretching the boundaries of this discipline.

REFERENCES

[1] C. P. Chen, and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314-347, 2014.

[2] W. Khallouli, and J. Huang, "Cluster resource scheduling in cloud computing: literature review and research challenges," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 6898-6943, 2022.

[3] W. Lin, G. Peng, X. Bian, S. Xu, V. Chang, and Y. Li, "Scheduling algorithms for heterogeneous cloud environment: main resource load balancing algorithm and time balancing algorithm," *Journal of Grid Computing*, vol. 17, pp. 699-726, 2019.

[4] F. M. Awaysheh, M. Alazab, S. Garg, D. Niyato, and C. Verikoukis, "Big data resource management & networks: Taxonomy, survey, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2098-2130, 2021.

[5] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, pp. 1-41, 2015.

[6] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big Data and cloud computing: innovation opportunities and challenges," *International Journal of Digital Earth*, vol. 10, no. 1, pp. 13-53, 2017.

[7] K. A. Alam, R. Ahmad, A. Akhunzada, M. H. N. M. Nasir, and S. U. Khan, "Impact analysis and change propagation in service-oriented enterprises: A systematic review," *Information Systems*, vol. 54, pp. 43-73, 2015.

[8] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing and Applications*, vol. 32, pp. 1531-1541, 2020.

[9] A. Senthil Kumar, and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Personal Communications*, vol. 107, pp. 1835-1848, 2019.

[10] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2370-2382, 2022.

[11] A. Hussain, and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 1-14, 2020.

[12] M. Huang, H. Han, H. Wang, L. Li, Y. Zhang, and U. A. Bhatti, "A clinical decision support framework for heterogeneous data sources," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1824-1833, 2018.

[13] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, pp. 1-17, 2014.

[14] C. Liu, K. Wu, and J. Pei, "An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 1010-1023, 2007.

[15] B. Louie, P. Mork, F. Martin-Sanchez, A. Halevy, and P. Tarczy-Hornoch, "Data integration and genomic medicine," *Journal of Biomedical Informatics*, vol. 40, no. 1, pp. 5-16, 2007.

[16] P. C. Stoy, M. Mauder, T. Foken, B. Marcolla, E. Boegh, A. Ibrom, M. A. Arain, A. Arneth, M. Aurela, and C. Bernhofer, "A data-driven analysis of energy balance closure across FLUXNET research sites: The role of landscape scale heterogeneity," *Agricultural and Forest Meteorology*, vol. 171, pp. 137-152, 2013.

[17] Y. Ling, "Design of 3D animation color rendering system supported by cloud computing based on genetic algorithm," *Soft Computing*, vol. 27, no. 14, pp. 10317-10326, 2023.

[18] G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments," *Future Generation Computer Systems*, vol. 110, pp. 1079-1097, 2020.

[19] H. Zhou, Y. Feng, and L. Han, "The hybrid heuristic genetic algorithm for job shop scheduling," *Computers & Industrial Engineering*, vol. 40, no. 3, pp. 191-200, 2001.

[20] T. F. Abdelmaguid, A. O. Nassef, B. A. Kamal, and M. F. Hassan, "A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles," *International Journal of Production Research*, vol. 42, no. 2, pp. 267-281, 2004.

[21] C. Veller, N. Kleckner, and M. A. Nowak, "A rigorous measure of genome-wide genetic shuffling that takes into account crossover positions and Mendel's second law," *Proceedings of the National Academy of Sciences*, vol. 116, no. 5, pp. 1659-1668, 2019.

[22] F. Herrera, M. Lozano, and A. M. Sánchez, "Hybrid crossover operators for real-coded genetic algorithms: an experimental study," *Soft Computing*, vol. 9, pp. 280-298, 2005.

[23] J. K. Cochran, S.-M. Horng, and J. W. Fowler, "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines," *Computers & Operations Research*, vol. 30, no. 7, pp. 1087-1102, 2003.

[24] S. Yuan, B. Skinner, S. Huang, and D. Liu, "A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms," *European Journal of Operational Research*, vol. 228, no. 1, pp. 72-82, 2013.

[25] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, pp. 2467-2501, 2016.