Data Science and Engineering (2026) 11:53–65
https://doi.org/10.1007/s41019-025-00314-w

RESEARCH PAPERS

A Cost-Saving Response Scheduler for Highway Structural Health Monitoring Data Applications

Zhixin Qi[1] · Yulin Wang[1] · Zemin Chao[1] · Zejiao Dong[1] · Hongzhi Wang[1]

Received: 20 February 2025 / Revised: 26 April 2025 / Accepted: 13 August 2025 / Published online: 13 November 2025

Abstract
In the process of responding to user applications on the highway structural health monitoring data sharing platforms, the objective is to decrease the network transmission costs and avoid a mass of redundant Input/Output operations between the storage server and local hard disks. Since none of existing job scheduling and structural health monitoring data analysis research has focused on this topic, we study the problem of cost-saving response scheduling for highway structural health monitoring data applications. To solve this problem, we develop a greedy response scheduler with $(1+\frac{1}{e-1})$-approximation ratio. Evaluation results demonstrate the effectiveness and efficiency of our proposed solution.

Keywords Monitoring data · Response scheduling · Data transmission · Approximation algorithm · Data sharing

1 Introduction

With the constant improvement of highway construction, it is an important task to monitor the long-term performance of highway service for structural health evaluation and maintenance decision making. As the cornerstone of revealing the essence of highway disease, structural health monitoring (SHM for brief) data are collected and stored by many regions and institutions. In recent years, massive highway monitoring data have been published online and an interactive interface is provided for users to submit applications of downloading and using data[1].

To obtain the transient information of highway structure under the high-speed vehicle loads, the sampling frequency of highway SHM data reaches up to 2000Hz. Such high frequency makes the data size reach 5 to 6 TB each month, which brings a heavy burden on the SHM data storage and management. Due to the limited storage capacity, local hard disk is used to store a small amount of SHM data. The remaining part of massive data is saved by storage server. To guarantee the information security, the storage server is not connected to the external network in case of server attacks and data loss. Therefore, the front-end server of highway monitoring data repository is only able to obtain data from storage server rather than store a great amount of data itself.

The heterogeneous data storage structure which consists of front-end server and storage server is commonly used in the domain of data service [1–4]. When users expect to download some data from the highway SHM data repository online, they have to submit a using application first. If the manager of data platform approves the application, the user-required data are compressed and sent to the user via a temporary link. Since the storage size of local disk is limited, it is possible that not all the needed data are stored in the local disk. In this case, we have to fetch the required data from storage server to local disk and then send them to

✉ Zhixin Qi
qizhx@hit.edu.cn

Yulin Wang
crystalniall@outlook.com

Zemin Chao
chaozm@hit.edu.cn

Zejiao Dong
hitdzj@hit.edu.cn

Hongzhi Wang
wangzh@hit.edu.cn

[1] Harbin Institute of Technology, Huanghe Road 73, Harbin, Heilongjiang, China

[1] https://www.roadsdata.cn/

the user. In addition, such scheduling may also be needed for user-required data in order to cope with real-time data access by users on the data sharing platform.

Furthermore, multiple servers are strategically placed across different geographical locations to handle applications closer to where the data is stored or needed, reducing latency and improving response times for data retrieval. In addition to optimizing application handling, this multi-server setup also plays a critical role in mitigating network congestion issues caused by frequent data fetching operations between local disks and storage servers.

However, the fetched data needs to take up local storage space which has been full with SHM data. Hence, a part of data in the local hard disk will be covered by the fetched data. Note that since the covered data have been stored in the storage server, there is no data loss even though the stored data in local storage space are covered. But the frequent fetching operations lead to the reduction of network Input/Output (I/O for brief) performance. In practice, the transmission speed of network I/O is an order of magnitude slower than that of hard disk I/O. Therefore, as the bottleneck of massive SHM data transmission, the network I/O costs need to be saved.

In the research field of SHM data analysis, most studies have focused on data processing for anomaly detection [5, 6], disease diagnosis [7, 8], and infrastructure performance prediction [9–15]. As shown in Fig. 1, in the process of responding to SHM data applications, the time spent on data scheduling using network I/O takes up as much as 87.8% of the total process time. However, none of the existing work has considered optimizing the I/O costs in this process, so we attempt to fill this gap in this paper.

To decrease the network transmission costs and avoid a mass of redundant I/O operations, we define the problem of cost-saving response scheduling for highway SHM data. The problem is proved as an NP-hard problem, which means that for modern computing machines, it is not guaranteed that an exact solution is obtained in practical time. To solve this problem, we design a cost-saving scheduler to determine the response order of the highway SHM data applications from users. Based on the storage states of local hard disk and storage server, the scheduler computes the overlap of the most recent user-required data sets. The data repository manager first responds to the application whose data overlap is the maximum. Via this solution, the I/O costs are reduced to at most $(1+\frac{1}{e-1})$-times the optimal ones, that is, if the optimal total I/O costs are 10TB, our solution creates at most 15.82TB.

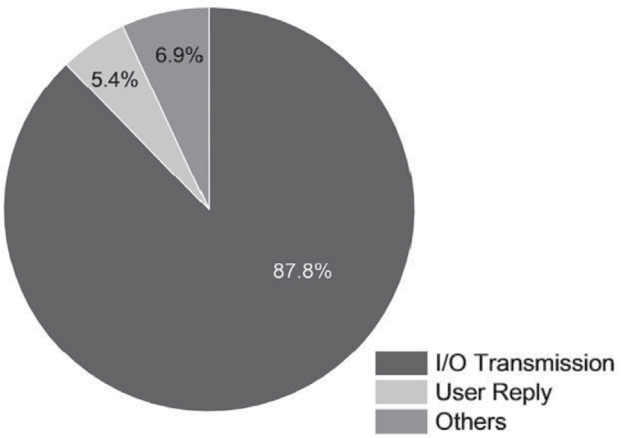


Fig. 1 Time overhead of responding to SHM data applications

To summarize, the contributions of this paper are as follows:

(i) We study the problem of cost-saving response scheduling for highway SHM data applications. To the best of our knowledge, this is the first work to optimize network I/O transmission costs in the research field of SHM data analysis.
(ii) We prove that the hardness of this problem is NP-hard and develop a greedy algorithm with $(1+\frac{1}{e-1})$-approximation ratio, which takes the overlap of SHM data into consideration.
(iii) Experimental evaluations demonstrate that our solution reduces the network I/O transmission costs up to average 42.16% compared with the most commonly used job scheduling strategies.

The rest of the paper is organized as follows. We first review the related work and provide the problem definition of our work. After that, we propose our solution and present the theoretical proof. Then, experimental results are discussed and analyzed. Finally, we conclude this paper.

2 Related Work

The related studies are divided into the following two main lines.

Job Scheduling. The definitions of job scheduling problem in different areas can be classified into single-objective optimization problem [16–22] and multi-objective optimization problem [23–28].

Aiming to minimize the maximum completion time, Nouiri et al. applied particle swarm optimization algorithm to solve the job scheduling problem [16]. To minimize total tardiness of jobs in the sequences, Azadeh et al. designed an integrated approach based on artificial neural network, genetic algorithm, and computer simulation to explore all the solution space in stochastic flexible flow shop with sequence-dependent setup times, job deterioration and

learning effects [17]. To minimize the makespan, Chen et al. presented a simple $\frac{5}{3}$-approximation algorithm based on a partition of the job set into agreeable layers using the natural layered representation of the directed acyclic precedence graph [18]. They also developed a greedy algorithm to reduce the number of singleton-job layers, resulting in an improved partition, which leads to a $\frac{4}{3}$-approximation algorithm. Chen et al. presented a self-learning genetic algorithm involving Q-learning for job scheduling [19]. The components of reinforcement learning were redesigned, including the state of genetic algorithm environment, the action of parameters, and the computing method of reward. To solve the complex structure problem in job scheduling, Park et al. proposed a new genetic algorithm based on a multi-chromosome, which involves two kinds of decisions: machine selection and operation sequencing [20]. To achieve the goal to minimize the makespan in job scheduling, Huang et al. constructed an auto mixed integer linear programming model [21]. The model computed the precedence between operations of a job to be given by an arbitrary directed acyclic graph rather than a linear order. Sun et al. designed an improved hybrid genetic algorithm with variable neighborhood search to address the job scheduling problem considering the machine workload balance in machining system, with the minimization of the makespan [22].

Kato et al. presented a solution of the multi-objective job scheduling, using a hierarchical approach that divides the problem into two sub-problems, that are the Particle Swarm Optimization to resolve the routing sub-problem and Random Restart Hill Climbing for the resolution of scheduling sub-problem [23]. Altoe et al. treated job scheduling in a multi-objective way with makespan and total tardiness as criteria [24]. They proposed a meta-heuristic algorithm based on the Clustering Search to produce a set of non-dominated solutions. Dai et al. defined a multi-objective optimization model with the objective to minimize energy consumption and makespan [25]. To solve this model, they developed an enhanced genetic algorithm. Tan et al. formulated a mixed-integer programming model with the objectives to minimize the total carbon emission and makespan [26]. They designed an enhanced multi-objective particle swarm optimization method to solve it. To address the bi-objective job scheduling problem with respect to minimization of the maximum completion time and total tardiness, Turkyilmaz et al. proposed an algorithm called bi-objective hybrid genetic algorithm which integrates genetic strategy with a multi-search algorithm and used hyper-volume contribution measure in its two-level selection strategy [27]. Facing the multi-objective optimization of stochastic failure-prone job scheduling problem, the goal of Amelian et al. was to determine the best sequence of jobs, optimal production rate, and optimum preventive maintenance period for simultaneous optimization of three criteria of sum of earliness and tardiness, system reliability, and energy consumption [28]. They first presented a new mixed integer programming model to formulate the problem. Then, by combining of simulation and NSGA-II algorithm, they put forward a new solution to solve this problem.

Comparing our work with the studies discussed above, we found that current methods mainly focused on the job scheduling problem in the industrial community. Facing the field of transportation infrastructure, our work defines the cost-saving response scheduling problem for highway SHM data applications, which is a kind of single-objective optimization problem.

SHM Data Analysis. Existing research analyze SHM data mainly for anomaly detection [5, 6], disease diagnosis [7, 8], and infrastructure performance prediction [9–15].

Bao et al. constructed a deep neural network for anomaly data classification to detect the potential anomalies in SHM data [5]. Tang et al. presented a new anomaly detection method based on convolutional neural network to detect the multi-pattern anomalies of SHM data [6].

Based on SHM data, Shang et al. designed a deep convolutional denoising auto-encoder to extract the damage features and detect the diseases for bridges [7]. Han et al. developed a bridge health monitoring system and utilized it to accomplish bridge health diagnosis [8].

Wang et al. proposed a Bayesian modeling approach to forecast the structural strain response using SHM data [9]. Based on real-time strain measurements, Kaloop et al. adopted double filtering and polynomial prediction to estimate the dynamic behavior of highway steel plate girder bridges [10]. Zhu et al. first leveraged tunnel suitability index to evaluate the shield tunnel status [11]. Then, they recognized different degradation models and predicted the performance of shield tunnel based on Long Short Term Memory model and Dynamic Time Wrapping approach. Sadeghian studied the reliability of ship structure using structural health monitoring data under two factors, fatigue and corrosion failure, and assessed the life of the structures [12]. Wang et al. proposed a method for local damage monitoring and global health evaluation based on pavement SHM data [13]. Using SHM data, Tan et al. developed a real-time prediction model to estimate the mechanical behaviors of underwater shield tunnel structure by coupling the spatio-temporal correlation with external load through auto-encoder network [14].

Du et al. discussed the framework of intelligent monitoring system for underground space infrastructure including anomaly detection, disease diagnosis, and performance forecasting modules, and reviewed the existing approaches in these domains [15].

While the above-mentioned work proposed a lot of effective methods to analyze SHM data for anomaly detection, disease diagnosis, and infrastructure performance prediction, none of current research considered the transmission costs of SHM data. To fill this gap, we take the network transmission I/O costs into account and develop a cost-saving response scheduler for highway SHM data.

Our work differentiates itself from all these existing studies in that the study of cost-saving response scheduling problem for highway SHM data applications is unexplored in previous work.

3 Problem Definition

In this section, we focus on the cost-saving response scheduling problem for highway SHM data applications.

Since our goal is to minimize the network I/O transmission costs of responding to user-required SHM data applications, we consider how to save expensive I/O costs with a reasonable response order. In order to avoid inputting or outputting repeated data sets, we involve data overlap in the evaluation of user-required data applications. Then, our problem is how to determine the response order with the consideration of data overlap.

The cost-saving response scheduling problem for highway SHM data applications is defined as follows.

Definition 1 (*Cost-saving response scheduling*) Given a time-ordered set $A = \{A_1, A_2,..., A_n\}$ of data applications, each application $A_i(1 \leq i \leq n)$ is a set of user-required data, the data size of A_i is not greater than the storage budget S of local hard disk, it spends network I/O transmission costs $C_i(1 \leq i \leq n)$ to respond to A_i, the cost-saving response scheduling problem determines the response order set A^* which minimizes the total I/O costs in the process of responding to A.

As defined above, we show the hardness of this problem as follows.

Theorem 1 *Cost-saving response scheduling for highway SHM data is an NP-hard problem.*

Proof We proof the NP-hardness of cost saving response scheduling problem by reducing the typical knapsack problem [29] to it in polynomial time.

Let $I = \{I_1, I_2,..., I_n\}$ denote the item set, $V = \{V_1, V_2,..., V_n\}$ be the value list of corresponding items, where $V_i(1 \leq i \leq n)$ is the value of item $I_i(1 \leq i \leq n)$, $W = \{W_1, W_2,..., W_n\}$ be the weight list of corresponding items, where W_i is the weight of item $I_i(1 \leq i \leq n)$, and W_c be the weight constraint of a knapsack. Then, an instance of knapsack problem with item set I, value set V, weight set W, and weight constraint W_c is formally expressed as $X = \{I, V, W, W_c\}$.

Given any instance $X = \{I, V, W, W_c\}$ of knapsack problem, we construct a corresponding instance Y of cost-saving response scheduling problem as follows.

Let $A = \{A_1, A_2,..., A_n\}$ denote the set of user applications, $C = \{C_1, C_2,..., C_n\}$ be the network I/O transmission cost list of corresponding user application, where $C_i(1 \leq i \leq n)$ is the transmission cost of application $A_i(1 \leq i \leq n)$, $D = \{D_1, D_2,..., D_n\}$ be the user-required data size list of corresponding application, where $D_i(1 \leq i \leq n)$ is the user-required data size of application $A_i(1 \leq i \leq n)$, and S be the data size budget of local hard disk. Finally, the instance of cost-saving response scheduling problem is formalized as $Y = \{A, C, D, S\}$.

Then, the solution of $X = \{I, V, W, W_c\}$ can be computed with the solution of $Y = \{A, C, D, S\}$ in polynomial time n.

In summary, the knapsack problem is reduced to the cost-saving response scheduling problem. Since it is well-known that the knapsack problem is NP-hard, the cost-saving response scheduling is also an NP-hard problem. □

Even though the hardness of cost-saving response scheduling problem is NP-hard, the solution for it has to be a polynomial approximation algorithm, which is suitable for a large number of user applications. To ensure the response order set given by the algorithm close to the optimal set, such approximation should have a low ratio bound. Thus, the challenge to solve this problem is to develop a polynomial approximate algorithm with a low ratio bound.

As discussed above, the optimization function F in cost-saving response scheduling problem is the sum of network I/O transmission costs in the process of responding to user applications. We first prove that F is an increasing submodular function via Lemma 1 and 2. The submodular property of F is shown in Lemma 1, and Lemma 2 shows the increasing property of F.

Lemma 1 *The optimization function F in cost-saving response scheduling problem is submodular. That is,*

$$F(A_a \cup A_x) - F(A_a) \geq F(A_b \cup A_x) - F(A_b)$$

when $A_a \subseteq A_b$*, where* $1 \leq a \leq n$*,* $1 \leq b \leq n$*, and* $1 \leq x \leq n$*.*

Lemma 1 means that given a user application set A_b and its subset A_a, when we add another application A_x into A_b and A_a, respectively, the increased optimization value of the new application set $A_b \cup A_x$ compared with A_b is not more than that of $A_a \cup A_x$ compared with A_a.

Proof Since $A_a \subseteq A_b$, the I/O cost of A_b is not less than that of A_a. Hence, when we append A_x to A_a, the possibility of involving new data sets is higher. It follows that:

Since

$$A_a \subseteq A_b,$$
$$(A_a \cup A_x) \setminus A_a \supseteq (A_b \cup A_x) \setminus A_b.$$

Accordingly,

$$F((A_a \cup A_x) \setminus A_a) \geq F((A_b \cup A_x) \setminus A_b).$$

Therefore,

$$F(A_a \cup A_x) - F(A_a) \geq F(A_b \cup A_x) - F(A_b).$$

This means that the increase of I/O costs is greater when A_x is appended to A_a compared to when we append A_x to A_b. □

Lemma 2 *The optimization function F is increasing. That is,*

$$F(A_a) \leq F(A_b)$$

when $A_a \subseteq A_b$*, where* $1 \leq a \leq n$ *and* $1 \leq b \leq n$*.*

Lemma 2 means that the optimization value of a user application set A_b is not less than that of its subset A_a.

Proof Since $A_a \subseteq A_b$, $A_a \cap A_b = A_a$. Hence, we have

$$F(A_a) = F(A_a \cap A_b)$$

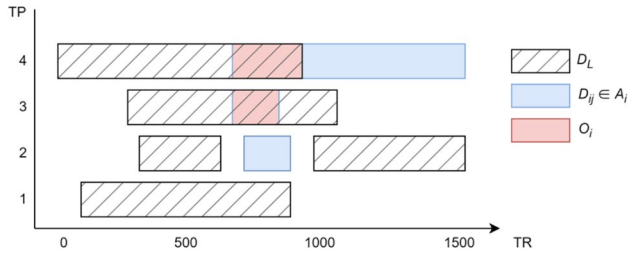


Fig. 2 Overlap definition example

and

$$F(A_b) = F(A_a \cap A_b) + F(A_b \setminus A_a).$$

Thus,

$$F(A_a) \leq F(A_b).$$

□

Now we show that the optimization function of cost-saving response scheduling problem is an increasing submodular function. The increasing submodular property drives the solution discussed next.

4 Proposed Solution

In this section, we develop a greedy approximation solution with the increasing submodularity of the optimization function to solve the cost-saving response scheduling problem. Our basic idea is to greedily respond to the user application which maximizes the data overlap compared with the data stored in local hard disk.

Before discussing the greedy solution, we first explain the computation process of data overlap. Each user application $A_i \in A$ ($1 \leq i \leq n$) consists of a set of required data, that is $A_i = \sum_{j=1}^{m} D_{ij}$. Since there are many different types of highway SHM data, such as transverse strain data, longitudinal strain data, vertical strain data, and vertical stress data, each $D_{ij} \in A_i$ contains a type label *TP*. As highway SHM data are collected with high frequency, there is also a time range label *TR* including the starting and ending time in each D_{ij}. Therefore, both the data which are needed by users and stored in local hard disk are labelled by different types and time ranges.

As depicted in the Fig. 2, based on the above discussion, we define the data overlap of a user application as follows.

Definition 2 Given a user application $A_i = \sum_{j=1}^{m} D_{ij}$ and the data in local hard disk currently D_L, the data overlap O_i of A_i is:

$$O_i = \forall TR_x \sum_{k=1}^{N} getSize(D_{ij}(TP_k, TR_x) \subseteq D_L(TP_k, TR))$$

where the function *getSize*() returns the size of a data set, $D_{ij}(TP_k, TR_x)$ denotes the data set D_{ij} whose type label is TP_k and time range label is TR_x.

We design our solution to cost-saving response scheduling problem based on the computation of data overlap. The main idea is to give the priority to the application whose data overlap with the data stored in local hard disk is maximum. Since the network transmission cost is directly proportional to the transmission data size and the transmission data size is negatively related to the overlap between user-required data and local data. Therefore, when the data overlap is maximum, the network I/O transmission costs are the least.

For example, consider a highway SHM data sharing platform where multiple user applications accumulate during the platform's application response cycle. Suppose that User A and User B each require a certain amount of data to complete their objects. Under the current state of the local disk storage, the data required by User A has a significant overlap with the local disk storage, while the data required by User B needs to be almost entirely scheduled from the storage server, i.e., the overlap of User A's required data is larger than that of User B's . In this case, our solution prioritizes the data scheduling of Application A to minimize the additional network I/O transmission cost.

The pseudo codes of our solution are shown in Algorithm 1. The inputs are the given time-ordered data application set *A* and the storage constraint *S* of local hard disk. The algorithm produces a response order set A^*.

We first initialize an empty set A^* to record the response order for the given set A (Line 1). Using the definition of data overlap discussed above, we compute the overlap O_i for each data application A_i in A (Lines 2–6). Then, the application A_x which has the maximum data overlap is responded and added to A^* (Lines 7–9). The process is repeated until all the data applications in A are responded (Line 10). Finally, we take the response order set A^* as the output (Line 11).

The time complexity of Algorithm 1 is determined by the number of data applications in *A* (denoted by *n*). The computation cost of the maximum data overlap for each A_x is *O*(*nlogn*) (Lines 4–8). The time cost of determining the response order set for all data applications in *A* is $O(n^2logn)$ (Lines 1–11). Therefore, the time complexity of Algorithm 1 is $O(n^2logn)$. In practice, since the data repository manager responds to data applications periodically, there are not many applications in *A*. Thus, the value of *n* is small and the time complexity of Algorithm 1 is low.

The space complexity of Algorithm 1 is *O*(*n*), where *n* is the number of data applications in *A* (denoted by *n*).

We show the approximation ratio of Algorithm 1 in Theorem 2.

Theorem 2 *Algorithm 1 is* $(1+\frac{1}{e-1})$*-approximation. That is, for the computed response order set* A^* *and an optimal response order set* $\hat{A}$*, we have*

Input: data application set $A=\{A_1, A_2, ..., A_n\}$, storage constraint S of local hard disk
Output: response order set A^*
1 $A^* \leftarrow \emptyset$;
2 **while** A *is* $\neg\emptyset$ **do**
3 $O \leftarrow \emptyset$;
4 **foreach** $A_i \in A$ **do**
5 $O_i \leftarrow \text{Overlap}(A_i, S)$;
6 $O \leftarrow O \cup \{O_i\}$;
7 $A_x \leftarrow \arg\max_{A_i} O_i$;
8 respond to A_x ;
9 $A^* \leftarrow A^* \cup \{A_x\}$;
10 $A \leftarrow A \setminus \{A_x\}$;
11 **return** A^*;

Algorithm 1 Greedy Algorithm

Fig. 3 Applying example of proposed response scheduler

$$F(A^*) \leq \left(1 + \frac{1}{e-1}\right) F(\hat{A})$$

where F is the optimization function of cost-saving response scheduling problem.

Proof We first prove that F is an increasing submodular function via Lemma 1 and 2. Then, based on the results about greedy algorithms and the minimization of increasing submodular functions confirmed by [30, 31], the optimization value, that is the network I/O transmission costs of A^* is at most $(1+\frac{1}{e-1})$-times I/O costs of the optimal solution, i.e. $\hat{A}$. □

As a massive highway monitoring data sharing platform, our proposed solution is able to assist the platform administrator in the process of handling and responding to user data usage applications. The general workflow is illustrated in Fig. 3. When the platform administrator handles data usage applications submitted by users within a certain period, the relevant data from these applications are fed into the response scheduler. The proposed solution can swiftly provide a reply scheme for the platform administrator, ensuring that the network I/O transmission costs during the application response process are minimized.

5 Experimental Evaluation

To verify the performance of our solution, we first conduct a set of effectiveness experiments on semantic and real SHM data applications compared with commonly-used job scheduling approaches. Then, we test the efficiency of our greedy algorithm. Finally, based on the comparison of some principle replacement strategies, we determine the replacement method of our solution.

Data Sets. In our experiments, we use real SHM data from Jilin Tonghua Highway. Data set information are shown in Table 1. In each highway monitoring data file, there is a data type label and a time range label corresponding to *TP* and *TR* in Definition 2, respectively.

Setup. All experiments are conducted on a computer with 13th Gen Intel(R) Core(TM) i7-13620H CPU@2.40GHz, 16GB memory. All algorithms are implemented in Python.

Table 1 Data Set Information

Data type	#-Sensors	Size of each file
Transverse strain	6	329.58MB
Longitudinal strain	10	549.30MB
Vertical strain	18	988.74MB
Vertical stress	9	494.40MB

5.1 Effectiveness Study

To test the effectiveness of our greedy solution in terms of reducing the network I/O transmission costs, we compare our approach (Ours for brief) with 8 commonly-used response strategies, that are response in application order (Order for brief), random response (Random for brief), response with the ascending order of user-required data size (AscSize for brief), response with the descending order of user-required size (DesSize for brief), response with the ascending order of the earliest timestamp in user-required data (AscTime for brief), response with the descending order of the latest timestamp in user-required data (DesTime for brief), response with the descending order of similarity to user-required data (Similar for brief), and response in the order generated based on Q-Learning (Q-Learn for brief).

Since the storage budget of local hard disk is 2TB, the size constraint of each data application is 2TB. Also, the timestamp of data application is accurate to hour. In the initial state, the local hard disk is full with four data types. The size of each data type is 0.5TB and the time range labels are the latest. When we replace the data stored in local hard disk with the fetched data from storage server, the random replacement strategy is adopted.

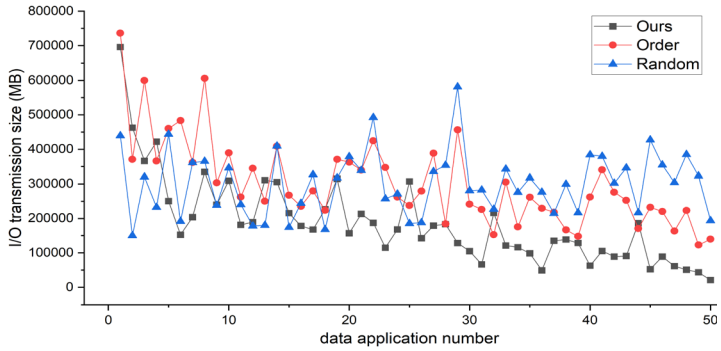
We first randomly generate a set of 50 data applications as the semantic data set. The number of data types, data type, data size of each type, time range in each application are all set at random. Since the network I/O transmission costs are directly proportional to the I/O transmission data size, we measure the transmission size of our method and the other 8 approaches. For fair comparison, we generate the parameters of each data application 10 times and report the average value of I/O transmission size. The comparison results are shown in Fig. 4.

As depicted in Fig. 4, Our method exhibits a decreasing trend of I/O transmission size in Ours as the number of data applications which are processed increases, while the I/O size of the other approaches fluctuates sharply. This is because that our solution selects the application whose overlap with the data stored in local hard disk is the largest each time.

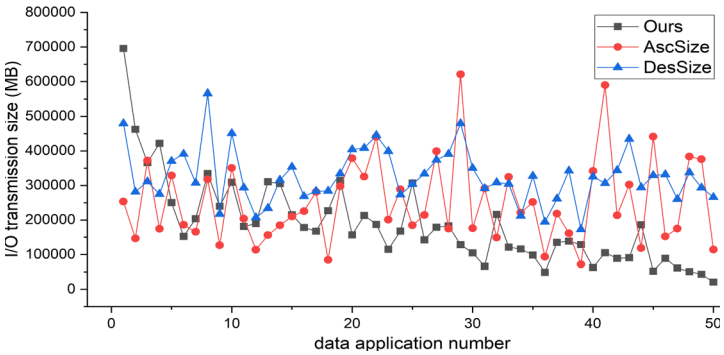
Then, we test the I/O transmission size on the real data set which consists of 50 data applications created by users. The evaluation results of our method and comparison approaches are shown in Fig. 5.

From Fig. 5, we observe that there is no I/O transmission size in Ours when responding to several data applications. This is because that when all the user-required data are stored in local hard disk, the network I/O costs are saved. And the total I/O transmission size on real data sets of Ours is the least among all the scheduling methods. The experimental results also indicate that, apart from Ours, Similar

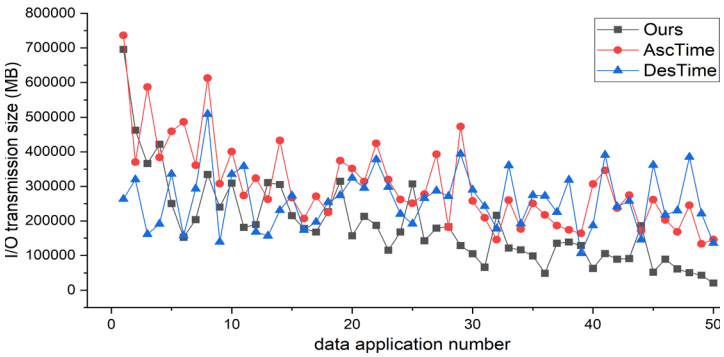
Fig. 4 Comparison results of response scheduling methods on semantic data sets



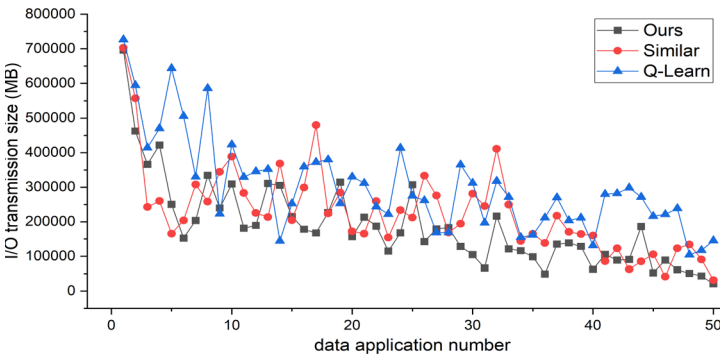
(a) Results of each application with Order and Random.



(b) Results of each application with AscSize and DesSize.

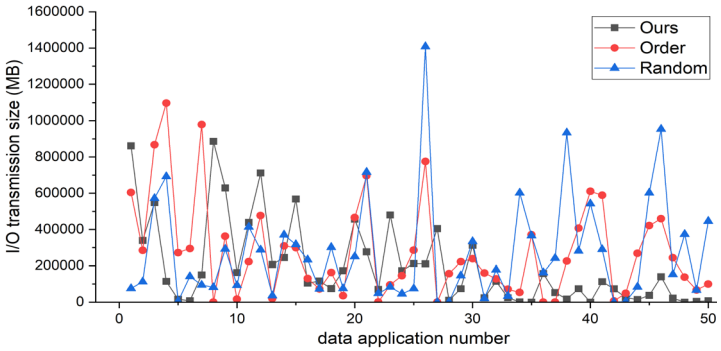


(c) Results of each application with AscTime and DesTime.

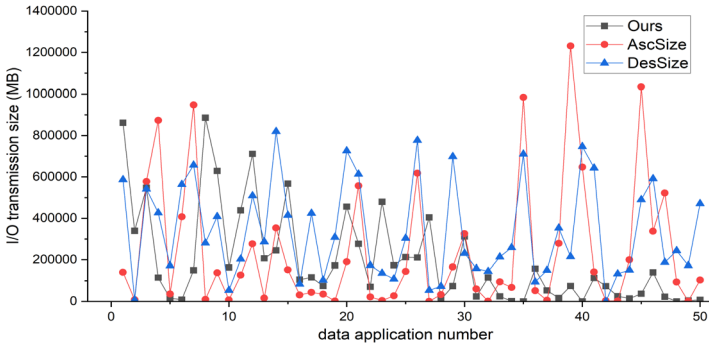


(d) Results of each application with Similar and Q-Learn.

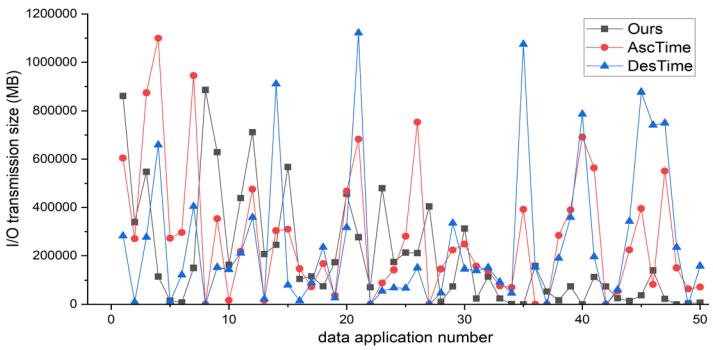
Fig. 5 Comparison results of response scheduling methods on real data sets



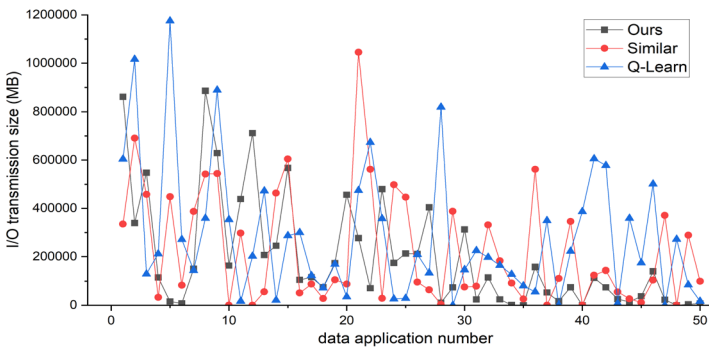
(a) Results of each application with Order and Random.



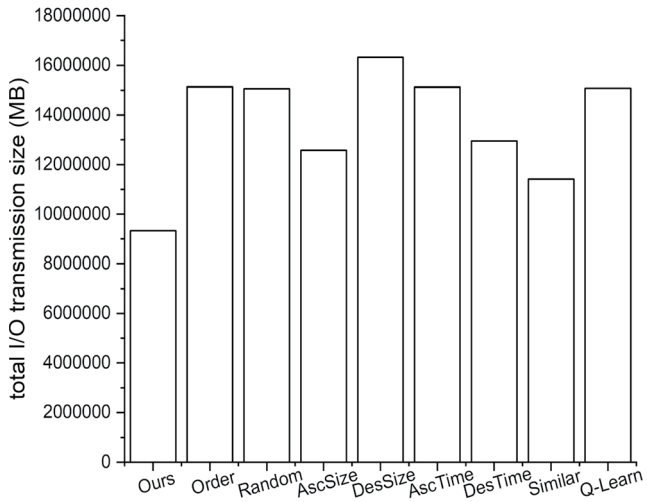
(b) Results of each application with AscSize and DesSize.



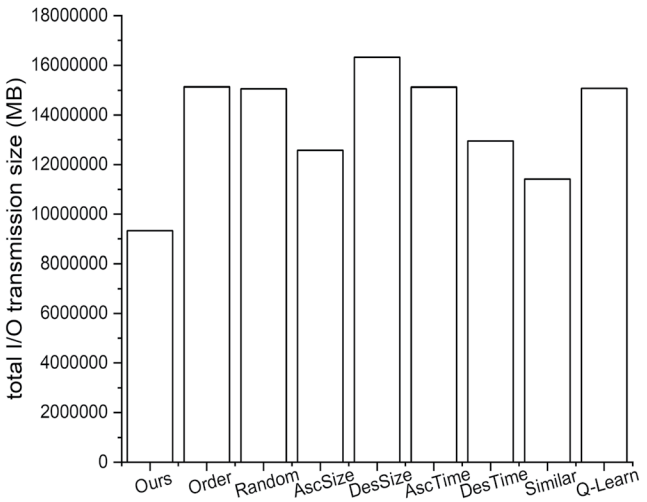
(c) Results of each application with AscTime and DesTime.



(d) Results of each application with Similar and Q-Learn.



(a) Total results on semantic data sets.



(b) Total results on real data sets.

Fig. 6 Total I/O transmission sizes of response scheduling methods

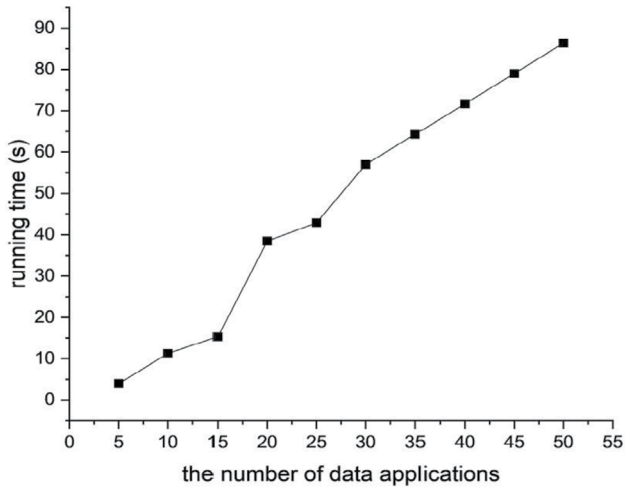


Fig. 7 Efficiency results varying the number of data applications

achieves the second-best performance. This is attributed to its utilization of the greedy principle, where it attempts to find the optimal solution at each step of the response process. However, due to the random nature of its first response, which introduces a cold-start phase, the effectiveness of Similar falls short in comparison to Ours. Additionally, due to the extremely large state space in this problem, the performance of Q-Learn based on reinforcement learning is not satisfactory.

In addition, the total I/O transmission sizes of response scheduling methods are show in Fig. 6. According to the evaluation results, our solution effectively reduces the network I/O transmission costs both on the semantic and the real data sets.

Considering the evaluation results on both semantic and real data sets, our solution reduces the network I/O transmission size up to average 42.16% compared with the most commonly used job scheduling approaches. This further confirms the effectiveness of our solution on saving network I/O transmission costs.

5.2 Efficiency Study

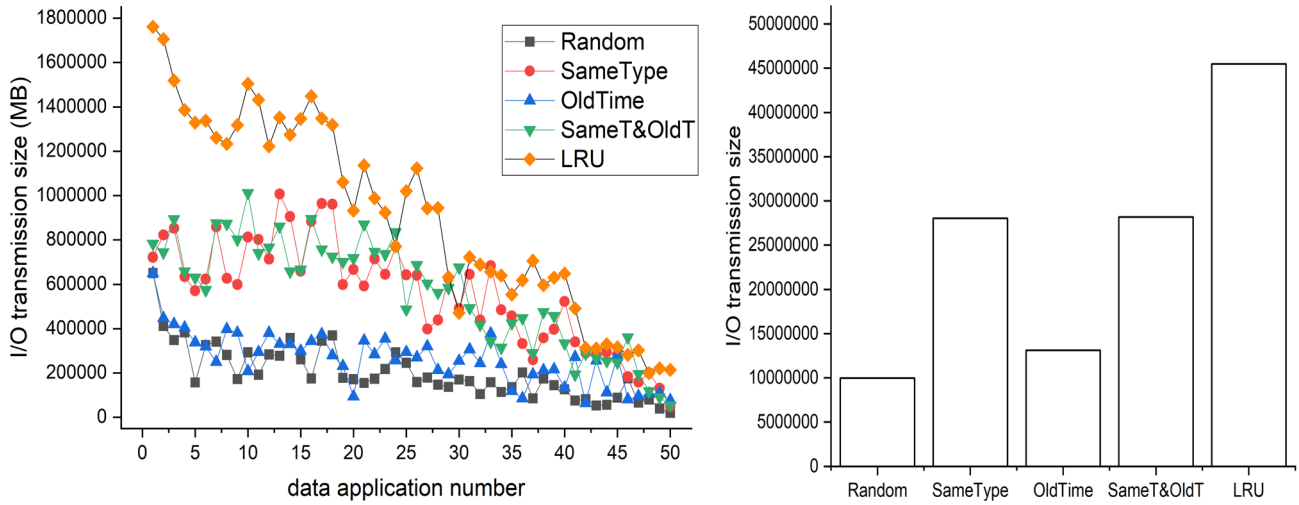
To test the efficiency of our solution, we measure the running time of Ours on 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 data applications, respectively. The experimental results are shown in Fig. 7.

As depicted in Fig. 7, we observe that as the number of data applications rises, the running time of Ours increases steadily. This trend is expected because the time costs of deciding which application is worthwhile processing first by Ours are positively related with the number of data applications. The results further demonstrate the time complexity of our greedy algorithm.

In practice, there are few data applications which wait for processing. Therefore, we are able to spend little running time to finish the process of deciding the response order of data applications.

5.3 Replacement Strategy Selection

When we replace the data currently stored in local hard disk with the fetched data from storage server, there are some candidate replacement strategies. To select the best replacement strategy for SHM data, we test the effectiveness of



(a) Comparison results of each application.

(b) Total comparison results.

Fig. 8 Comparison results of different replacement strategies

different strategies, that are replacing SHM data randomly (Random for brief), replacing SHM data with the same type first (SameType for brief), replacing SHM data with the oldest timestamp first (OldTime for brief), replacing SHM data with the same type and the oldest timestamp first (SameT&OldT for brief), and replacing the least recently used SHM data first (LRU for brief), in our solution. The experimental results are depicted in Fig. 8.

From Fig. 8, we observe that as the number of data applications which are processed increases, there is a descending trend in all replacement strategies. This is due to the fact that our solution selects the application whose overlap with the data stored in local hard disk is the largest each time. Also, this confirms that the effectiveness is not affected by the type of replacement strategy. For the LRU strategy, due to the lack of historical applications in the early stages, it suffers significantly from the impact of data unscheduled visit and long-term periodicity in the preliminary phases of application response. This results in its substantial I/O transmission costs. In addition, in most cases, the I/O transmission size of Random is lower than the other strategies and the total I/O size of Random is the least. This is because that there is not a fixed pattern in user-required data applications. The data type and data timestamp are set at random by users in the process of applying for SHM data. Therefore, we recommend adopting Random replacement strategy when using our approach.

In summary, the evaluation results demonstrate that our proposed solution with the approximation algorithm solves the cost-saving response scheduling problem for highway SHM data applications effectively and efficiently.

6 Conclusion

In this paper, we proposed a novel cost-saving response scheduler for highway SHM data applications. Based on the scheduler, the response order is determined by data overlap greedily in the process of responding to data applications. Experimental results show that our solution decreases the network I/O transmission size up to average 42.16% compared with the most commonly used job scheduling approaches.

Since the highway SHM data are often accessed and required by users, our future work aims to present visual display of SHM data on the data sharing platforms. In addition, since it is difficult for users to determine the useful data, we are also interested to provide the interfaces of SHM data sketch to improve the user experience.

Representative Publications of First Author

1. Qi Z, Wang H, Dong Z. Dirty Data Processing for Machine Learning[M]. Springer, 2024.
2. Qi Z*, Wang H. Access Structure Selection for Knowledge Graphs Based on Machine Learning[C]//ACM Turing Award Celebration Conference 2024. 2024: 214-215.
3. Qi Z, Wang H*, Zhang H. A Dual-Store Structure for Knowledge Graphs[J]. IEEE Transactions on Knowledge & Data Engineering, 2023, 35(02): 1104-1118.

4. Qi Z, Wang H*, Shen Z, et al. PreKar: A learned performance predictor for knowledge graph stores[J]. World Wide Web, 2023, 26(1): 321-341.
5. Qi Z, Zhang H, Wang H*, et al. ANSWER: Automatic Index Selector for Knowledge Graphs[C]//Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data. Singapore: Springer Nature Singapore, 2023: 393-407.
6. Wang C, Wang H*, Mu T, Qi Z, et al. Evaluating community quality based on ground-truth[J]. Information Sciences, 2022, 599: 104-126.
7. Yan Y, Wang H*, Wang Y, Qi Z, et al. Multi-sql: An automatic multi-model data management system[C]// Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data. Cham: Springer Nature Switzerland, 2022: 451-455.
8. Qi Z X, Wang H Z*, Wang A J. Impacts of dirty data on classification and clustering models: an experimental evaluation[J]. Journal of Computer Science and Technology, 2021, 36: 806-821.
9. Qi Z, Wang H*. Dirty-data impacts on regression models: an experimental evaluation[C]//Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11¨C14, 2021, Proceedings, Part I 26. Springer International Publishing, 2021: 88-95.
10. Qi Z, Wang H*, He T, et al. TAILOR: time-aware facility location recommendation based on massive trajectories[J]. Knowledge and Information Systems, 2020, 62(9): 3783-3810.
11. Qi Z, Wang H*, He T, et al. Friend: Feature selection on inconsistent data[J]. Neurocomputing, 2020, 391: 52-64.
12. Wang H*, Qi Z, Zheng L, et al. April: An automatic graph data management system based on reinforcement learning[C]//Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020: 3465-3468.
13. 齐志鑫, 王宏志*, 周雄, 等. 劣质数据上代价敏感决策树的建立[J]. 软件学报, 2019, 30(3): 604-619.
14. Qi Z, Wang H*, Li J, et al. FROG: Inference from knowledge base for missing value imputation[J]. Knowledge-Based Systems, 2018, 145: 77-90.
15. Qi Z, Wang H*, Meng F, et al. Capture missing values with inference on knowledge base[C]//Database Systems for Advanced Applications: DASFAA 2017 International Workshops: BDMS, BDQM, SeCoP, and DMMOOC, Suzhou, China, March 27-30, 2017, Proceedings 22. Springer International Publishing, 2017: 185-194.
16. Wang H Z*, Qi Z X, Shi R X, et al. COSSET+: Crowdsourced missing value imputation optimized by knowledge base[J]. Journal of Computer Science and Technology, 2017, 32: 845-857.
17. Zheng K, Wang H*, Qi Z, et al. A survey of query result diversification[J]. Knowledge and Information Systems, 2017, 51: 1-36.
18. Zhang L*, Zhou T, Qi Z, et al. The research on e-mail Users' behavior of participating in Subjects based on social network analysis[J]. China Communications, 2016, 13(4): 70-80.

Author Contributions Zhixin Qi: writing original draft, methodology; Yulin Wang: data curation, experiments; Zemin Chao: theoretical proof; Zejiao Dong: supervision; Hongzhi Wang: project administrator.

Funding This paper is supported by the National Key Research and Development Program of China (2024YFE0214600) and the National Natural Science Foundation of China (52308449).

Data Availibility The data that support the findings of this study are partially available from the corresponding author upon reasonable requests.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Kumar R, Grot B (2022) Shooting down the server front-end bottleneck. ACM Trans Comput Syst (TOCS) 38(3–4):1–30
2. Deng F, Cao Q, Wang S, Liu S, Yao J, Dong Y, Yang P (2020) Serw: Adaptively separating read and write upon ssds of hybrid storage server in clouds. In: Proceedings of International Conference on Parallel Processing, pp. 1–11
3. Chen Y, Zhao T, Cheng P, Ding M, Chen CW (2022) Joint front-edge-cloud iovt analytics: resource-effective design and scheduling. IEEE Internet Things J 9(23):23941–23953
4. Chiang M-L, Hou T-T (2020) A scalable virtualized server cluster providing sensor data storage and web services. Symmetry 12(12):1942

5. Bao Y, Tang Z, Li H, Zhang Y (2019) Computer vision and deep learning-based data anomaly detection method for structural health monitoring. Struct Health Monit 18(2):401–421
6. Tang Z, Chen Z, Bao Y, Li H (2019) Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring. Struct Control Health Monit 26(1):2296
7. Shang Z, Sun L, Xia Y, Zhang W (2021) Vibration-based damage detection for bridges by deep convolutional denoising autoencoder. Struct Health Monit 20(4):1880–1903
8. Han C, Li X, Fu X, Gu L, Ouyang Z et al (2021) Study on bridge structure damage and health diagnosis method based on health monitoring. Tehnički vjesnik 28(3):746–753
9. Wang Y, Ni Y (2020) Bayesian dynamic forecasting of structural strain response using structural health monitoring data. Struct Control Health Monit 27(8):2575
10. Kaloop MR, Elbeltagi E, Hu JW (2020) Estimating the dynamic behavior of highway steel plate girder bridges using real-time strain measurements. Appl Sci 10(12):4215
11. Zhu H, Wang X, Chen X, Zhang L (2020) Similarity search and performance prediction of shield tunnels in operation through time series data mining. Autom Constr 114:103178
12. Sadeghian M (2022) The reliability assessment of a ship structure under corrosion and fatigue, using structural health monitoring. Int J Eng 35(09):1765–1778
13. Wang N, Han T, Cheng H, Li T, Fu J, Ma T, Fu Y, Chen F, Zhang Y (2022) Monitoring structural health status of asphalt pavement using intelligent sensing technology. Constr Build Mater 352:129025
14. Tan X, Chen W, Zou T, Yang J, Du B (2023) Real-time prediction of mechanical behaviors of underwater shield tunnel structure using machine learning method based on structural health monitoring data. J Rock Mech Geotech Eng 15(4):886–895
15. Du B, Ye J, Zhu H, Sun L, Du Y (2023) Intelligent monitoring system based on spatio-temporal data for underground space infrastructure. Eng 25(6):194–203
16. Nouiri M, Bekrar A, Jemai A, Niar S, Ammari AC (2018) An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J Intell Manuf 29:603–615
17. Azadeh A, Goodarzi AH, Kolaee MH, Jebreili S (2019) An efficient simulation-neural network-genetic algorithm for flexible flow shops with sequence-dependent setup times, job deterioration and learning effects. Neural Comput Appl 31:5327–5341
18. Chen Y, Goebel R, Lin G, Su B, Zhang A (2020) Open-shop scheduling for unit jobs under precedence constraints. Theoret Comput Sci 803:144–151
19. Chen R, Yang B, Li S, Wang S (2020) A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput Ind Eng 149:106778
20. Park J-S, Ng H-Y, Chua T-J, Ng Y-T, Kim J-W (2021) Unified genetic algorithm approach for solving flexible job-shop scheduling problem. Appl Sci 11(14):6454
21. Huang L, Su R (2022) An auto-milp model for flexible job shop scheduling problem. IFAC-PapersOnLine 55(3):137–142
22. Sun K, Zheng D, Song H, Cheng Z, Lang X, Yuan W, Wang J (2023) Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system. Expert Syst Appl 215:119359
23. Kato ERR, Aranha GDA, Tsunaki RH (2018) A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and random-restart hill climbing. Comput Ind Eng 125:178–189
24. Altoé WA, Bissoli DdC, Mauri GR, Amaral AR (2018) A clustering search metaheuristic for the bi-objective flexible job shop scheduling problem. In: Latin American Computer Conference (CLEI), pp. 158–166
25. Dai M, Tang D, Giret A, Salido MA (2019) Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robot Comput-Integr Manuf 59:143–157
26. Tan W, Yuan X, Huang G, Liu Z (2021) Low-carbon joint scheduling in flexible open-shop environment with constrained automatic guided vehicle by multi-objective particle swarm optimization. Appl Soft Comput 111:107695
27. Türkyılmaz A, Senvar O, Ünal İ, Bulkan S (2022) A hybrid genetic algorithm based on a two-level hypervolume contribution measure selection strategy for bi-objective flexible job shop problem. Comput Oper Res 141:105694
28. Amelian SS, Sajadi SM, Navabakhsh M, Esmaelian M (2022) Multi-objective optimization for stochastic failure-prone job shop scheduling problem via hybrid of NSGA-II and simulation method. Expert Syst 39(2):12455
29. Pferschy U, Schauer J, Thielen C (2021) Approximating the product knapsack problem. Optim Lett 15(8):2529–2540
30. Bach F (2019) Submodular functions: from discrete to continuous domains. Math Program 175:419–459
31. Qi Z, Wang H, He T, Wang C, Li J, Gao H (2020) Tailor: time-aware facility location recommendation based on massive trajectories. Knowl Inf Syst 62(9):3783–3810