BenchCouncil Transactions on Benchmarks, Standards and Evaluations

TBench
Volume 4, Issue 1
2024

Original Articles

ISSN: 2772-4859

Contents

BenchCouncil Transactions on Benchmarks, Standards and Evaluations 4 (2024) 100161

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Full length article

TensorTable: Extending PyTorch for mixed relational and linear algebra pipelines

Xu Wen [*]

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:
Linear algebra
Relational algebra
Runtime system

ABSTRACT

The mixed relational algebra (RA) and linear algebra (LA) pipelines have become increasingly common in recent years. However, contemporary widely used frameworks struggle to support both RA and LA operators effectively, failing to ensure optimal end-to-end performance due to the cost of LA operators and data conversion. This underscores the demand for a system capable of seamlessly integrating RA and LA while delivering robust end-to-end performance. This paper proposes TensorTable, a tensor system that extends PyTorch to enable mixed RA and LA pipelines. We propose TensorTable as the unified data representation, storing data in a tensor format to prioritize the performance of LA operators and reduce data conversion costs. Relational tables from RA, as well as vectors, matrices, and tensors from LA, can be seamlessly converted into TensorTables. Additionally, we provide TensorTable-based implementations for RA operators and build a system that supports mixed LA and RA pipelines. We implement TensorTable on top of PyTorch, achieving comparable performance for both RA and LA operators, particularly on small datasets. TensorTable achieves a 1.15x-5.63x speedup for mixed pipelines, compared with state-of-the-art frameworks—AIDA and RMA.

1. Introduction

In recent years, mixed pipelines that integrate both relational algebra (RA) and linear algebra (LA) operators have become increasingly prevalent in fields like data science, artificial intelligence, and real-time analysis. For instance, analytical queries [1–4], which heavily rely on RA operators, leverage LA operators such as matrix multiplication for statistical computations. Meanwhile, machine learning pipelines [5–7], primarily built upon LA operators, utilize RA operators such as join for preprocessing. Real-time tasks [8–10] frequently switch between RA and LA operators to facilitate swift data processing and analysis. However, as shown in Fig. 2, currently widely-used frameworks are unable to support both RA and LA operators while ensuring optimal performance. RA systems [11–13] lack the optimizations for LA operators while LA systems [14–16] do not offer adequate support for RA operators. Cross-framework implementations [17–20] bring extra costs due to data copying and transformations between frameworks and limit optimizations.

The need arises for a system capable of seamlessly integrating RA and LA while delivering robust performance. Many previous works [17,18,21–30] attempt to address this issue. However, most of them prioritize the performance of RA operators but have high execution costs on LA operators and data conversion, thus leading to poor end-to-end performance. Our experiments shown in Fig. 3 corroborate this point. To tackle this problem, we aim to develop a system that seamlessly integrates mixed pipelines and provides a good end-to-end performance. We prioritize ensuring the performance of LA operators while supporting RA operators with comparable performance and reducing frequent data conversion.

Unfortunately, it is not a trivial task due to the inherent disparities between RA and LA. Specifically, RA and LA reflect distinct characteristics from the perspectives of data abstraction, data types, and implementation. In terms of data abstraction, RA operates on relational tables, whereas LA deals with vectors, matrices, and tensors. Regarding data types, RA accommodates both numerical and non-numerical data, whereas LA exclusively handles numerical data. From the perspective of implementation, to ensure the optimal performance of LA operators, frameworks should make full use of parallelism, keep good temporal and spatial locality, and utilize extended instructions properly. However, whether relational tables used in RA systems or DataFrames and RDDs used in general-purpose systems often fall short in guaranteeing optimal locality and instruction utility. Consequently, these frameworks tend to exhibit suboptimal performance when dealing with LA operators.

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
E-mail address: wenxu@ict.ac.cn.

https://doi.org/10.1016/j.tbench.2024.100161
Received 18 December 2023; Received in revised form 17 January 2024; Accepted 7 February 2024
Available online 7 March 2024

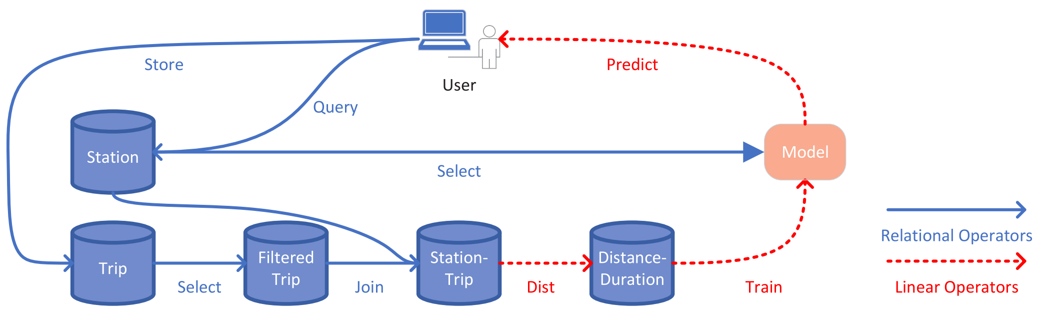


Fig. 1. A public bicycle sharing system leveraging mixed relational algebra (RA) and linear algebra (LA) pipelines to forecast user ride duration.

To prioritize ensuring the performance of LA operators, support RA operators with comparable performance, and reduce data conversion costs, we propose TensorTable as the unified abstraction for both RA and LA, which stores data in a tensor format. For LA, we directly encapsulate vectors, matrices, and tensors within TensorTable. For relational tables used in RA, we convert those non-numerical data into numerical data and use auxiliary dictionaries to preserve the mapping relations, subsequently storing the data in a tensor format. We provide TensorTable-based implementations for RA operators, covering selection, projection, join, group by, and aggregation. We establish a system capable of supporting the mixed pipelines of RA and LA, built on top of a typical LA system—PyTorch. Our work supports both RA and LA operators and achieves comparable performance.

This paper makes the following three contributions:

- We propose an abstraction—TensorTable, to represent both relational and linear algebra.
- We present TensorTable-based implementations for relational algebra operators and achieve comparable performance, especially for small datasets.
- We build a system for combining relational and linear algebra and achieve a 1.15x-5.63x speedup on mixed pipelines, compared with state-of-the-art frameworks—AIDA and RMA.

The remainder of the paper is organized as follows. Section 2 shows the background, motivation, and challenges. Section 3 shows the system overview. Section 4 introduces the design and implementation. Section 5 presents our evaluation. Section 6 illustrates the related work. Finally, we draw a conclusion in Section 7.

2. Background, motivation, and challenges

2.1. Background

Mixed relational algebra (RA) and linear algebra (LA) pipelines are becoming more and more common in recent years. Analytical queries [1–4], which heavily rely on RA operators, leverage LA operators such as matrix multiplication for statistical computations. Meanwhile, machine learning pipelines [5–7], primarily built upon LA operators, utilize RA operators such as join for preprocessing. Real-time tasks [8–10] frequently switch between RA and LA operators to facilitate swift data processing and analysis. Fig. 1 illustrates a typical real-time system—public bicycle sharing system [31] as an example. It leverages mixed LA and RA pipelines to forecast user ride duration and is a latency-sensitive task that requires optimal end-to-end performance. $Station$ stores station information and $Trip$ stores user trip history. The system uses RA operators such as select and join to process $Station$ and $Trip$, then uses LA operators such as calculate Euclidean distance and linear regression to train the predicting model. When users query the system, it uses RA operators such as select and LA operators such as linear regression to forest ride duration. Finally, new trip records are stored in $Trip$. This system integrates both RA and LA operators and requires optimal end-to-end performance.

2.2. Motivation

Despite the popularity of mixed pipelines, contemporary widely-used frameworks cannot support those pipelines and guarantee performance. As demonstrated in Fig. 2, our experiments encompass four typical relational operators – selection, projection, inner join, and group by, and two typical linear operators – matrix covariance and linear regression. Our findings reveal that LA systems like PyTorch cannot support many RA operators such as join and group by. Meanwhile, RA systems like MonetDB exhibit limited support for LA operators with subpar performance. General-purpose systems like Spark and pandas can handle both RA and LA operators, but their performance falls short of the ideal. Cross-framework implementations bring extra costs due to data copying and transformations between frameworks and limit optimizations. The last two columns in Fig. 2 display the conversion time from Spark DataFrame to Matrix and from pandas DataFrame to Tensor, respectively. This process incurs a substantial cost, even 6.7x-18.9x larger than that of RA operators over the same data sizes.

Additionally, we analyze the mixed pipeline introduced in Section 2.1, Fig. 3 shows the performance breakdown. We find that many previous works prioritize the performance of RA operators but have high execution costs on LA operators and data conversion, thus leading to poor end-to-end performance. Hence, there is a pressing need for a system capable of supporting mixed RA and LA pipelines, ensuring the performance of LA and RA operators and reducing data conversion costs, finally delivering optimal end-to-end performance.

2.3. Challenges

There are several challenges to overcome.

(1) Data abstraction. Data abstraction should fit the memory access patterns of its computation. RA computations are primarily based on column-based access, and some computations also rely on row-based access. LA computations are more complex, involving row-based access, column-based access, block-based access, and stride-based access. New data abstraction needs to be compatible with the different memory access patterns mentioned above. RA operators handle both numerical and non-numerical data, whereas LA operators exclusively consist of numerical data. New data abstraction should be able to handle both numerical and non-numerical data. The question that arises is: What constitutes the appropriate data abstraction for such mixed pipelines?

(2) Expressiveness. Mixed pipelines contain RA operators including selection, projection, join, group by, and aggregation as well as LA operators such as matrix multiplication. Existing algorithms may not be suitable for the new data abstraction. In this scenario, it is essential to propose new algorithms tailored to effectively express RA and LA operators.

(3) Performance. Many mixed pipelines are latency-sensitive tasks, which have high-performance requirements. A proper data abstraction that can fit the memory access patterns of its computation is necessary but not sufficient for achieving optimal performance. We should also consider hardware-related and dataflow graph optimizations, as well as data conversion costs.

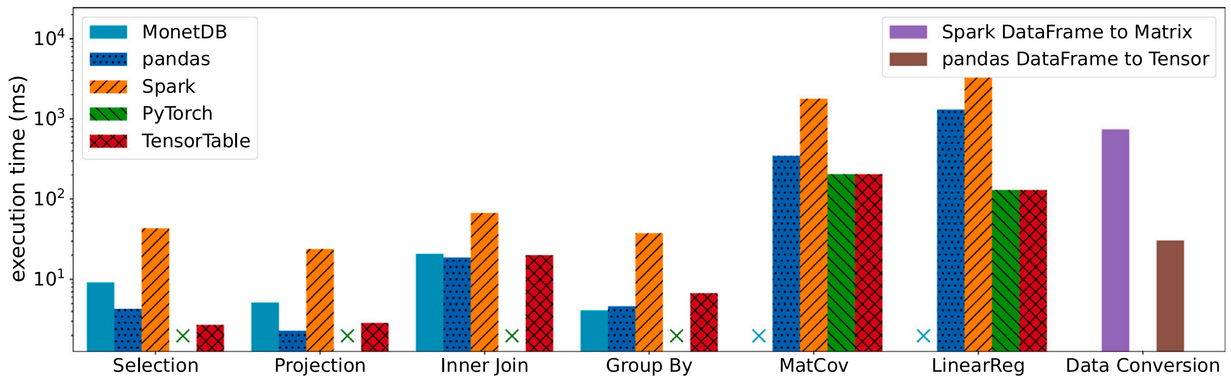


Fig. 2. The performance of popular frameworks on LA operators, RA operators, and data conversion. MatCov and LinearReg are short for matrix covariance and linear regression, respectively. "x" means unsupported. Currently widely-used frameworks are unable to support both RA and LA operators while ensuring optimal performance. Cross-framework implementations face large data conversion costs.

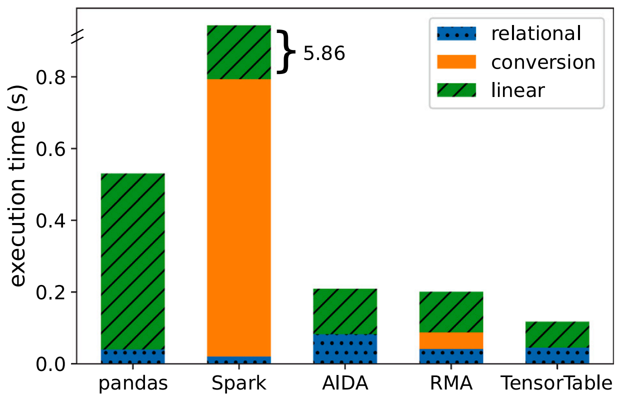


Fig. 3. The performance breakdown of the mixed pipeline in the bicycle sharing system. Many previous works have high execution costs on LA operators and data conversion, thus leading to poor end-to-end performance.

3. The system overview

This section shows the overview of our system.

3.1. Design choices

First, we analyze our design choices.

(1) The core objective is to ensure end-to-end performance for mixed pipelines. As analyzed in Section 2.2, many previous works have high execution costs on LA operators and data conversion, thus leading to poor end-to-end performance. Thus we prioritize ensuring the performance of LA operators, then support RA operators with comparable performance and reduce data conversion costs.

(2) Store all data in tensor formats. Tensors, with their data stored contiguously in the same memory block, enable access to data by computing offsets within the continuous memory block. This characteristic makes tensors suitable for accommodating all memory access patterns used in RA and LA computations, including row-based, column-based, block-based, and stride-based access. Besides, LA operators benefit significantly from key factors such as good parallelism, good temporal and spatial locality, and proper utilization of extended instructions—all of which are inherently tied to tensor abstractions. As a solution, we introduce a tensor-based abstraction called TensorTable, designed to represent both tensors and relational tables, while encoding non-numeric data into numeric representations.

(3) Utilize existing implementations and optimizations for LA operators and propose compatible algorithms and implementations for RA operators. We encapsulate LA operators without modifying their core functionality and supplement them with auxiliary functions to enable their seamless integration within mixed pipelines, not influencing existing hardware-related and dataflow graph optimizations. To support mixed pipelines, we offer TensorTable-based algorithms and implementations for RA operators, successfully achieving comparable performance without compromising the performance of LA operators.

(4) Perform data conversion during initialization instead of runtime to reduce data conversion costs. Frequent data conversion during runtime markedly impacts performance and should be minimized. The only necessary data conversion occurs during initialization, where we transform the original data into TensorTables. No additional data conversions are needed throughout the execution phase. To ensure optimal performance, each operator within mixed pipelines takes TensorTables as input and produces them as output without the need for extra data conversion.

3.2. System architecture

Fig. 4 illustrates the architecture of our system. We utilize TensorTable as the unified abstraction, as defined in Section 4.1. We convert the origin data to TensorTable during initialization and store them in memory. We propose Directed Acyclic Graph (DAG) Intermediate Representation(IR) to represent mixed pipelines, as detailed in Section 4.3.1. The Parser is responsible for translating mixed pipelines into DAG IRs, as elaborated in Section 4.3.2. The Optimizer makes graph-level optimizations, as discussed in Section 4.3.3. The Code Generator generates TensorTable-based operators, as shown in Section 4.3.4. For LA operators, we call PyTorch operators, while for RA operators, we provide TensorTable-based implementations, as defined in Section 4.2. Finally, those operators are executed on top of PyTorch Runtime, as described in Section 4.3.5.

4. The system design and implementation

4.1. The data abstraction: TensorTable

This section introduces our proposed abstraction—TensorTable. TensorTable represents relational tables in a tensor format, allowing for seamless processing. By taking TensorTables as both input and output, all RA and LA operators can be efficiently implemented on tensors. There are two notable benefits to storing data in a tensor format: Firstly, tensors exhibit better data locality compared to row-oriented and column-oriented tables, particularly benefiting LA operators. Secondly, tensors can leverage hardware features and compilation optimizations more effectively. TensorTable seamlessly encapsulates vectors, matrices, and tensors for LA operations, eliminating the need for

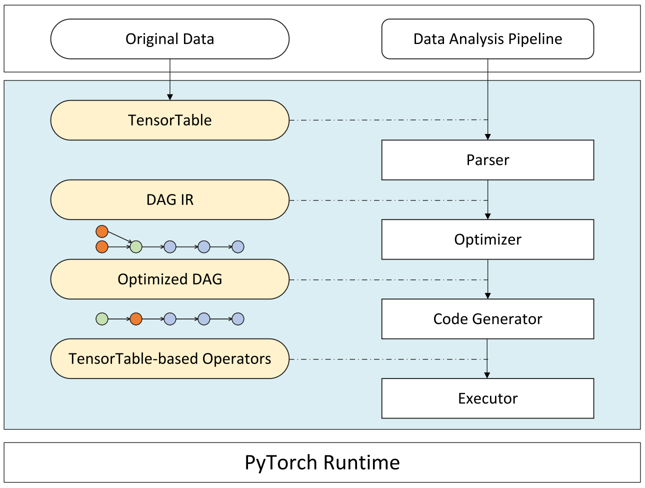


Fig. 4. The TensorTable Framework.

additional data conversions. As for RA operators, we convert relational tables to TensorTables and furnish them with TensorTable-based implementations.

TensorTable comprises four essential elements: *column_names*, *column_types*, *data_tensor*, and *str_dict_list*. Among these, *column_names* and *column_types* are string lists used for storing column names and data types, respectively. All data, whether numeric or non-numeric, is stored within the tensor *data_tensor*. Numeric data from relational tables can be directly stored in *data_tensor*. For non-numeric data, TensorTable encodes them into numeric representations and then stores them in the same tensor. Non-numeric data in relational tables fall into four categories: *date*, *boolean*, *string*, and *text*. For *date* data, we convert them into timestamps, represented as integers. *Boolean* data is converted into 0 or 1. For *string* and *text* data, we encode them as integers and maintain mapping relationships in dictionaries referred to as *str_dicts*.

String is usually used to represent categories, and RA operators such as join can operate on them. For *string* data, we sort them, get their unique elements, and then convert them to integers. As for *text* data, which typically has larger lengths, we map them directly to integers based on their order. Note that *str_dicts* are utilized solely for storing the mapping relations and preserving data order after complex relational transformation, without sophisticated processing such as embedding. Each string column maintains its own *str_dict*, while *str_dict_list* is a list containing these *str_dicts*, with its length corresponding to the number of string columns.

Fig. 5 showcases a TensorTable structure. The left in Fig. 5 shows the original relational table, with m rows and n columns. The right in Fig. 5 presents the corresponding TensorTable. Both *column_names* and *column_types* have a length of n, while *data_tensor* has a shape of m×n. Numeric data, such as that in the *code*, *latitude*, and *longitude* columns, is directly stored in *data_tensor*. The *name* column contains string data, which we map to integers. The mapping relation is retained in *str_dict1*. As this relational table has only one string column, the corresponding TensorTable has just one *str_dict*, namely *str_dict1*, and the *str_dict_list* consists of a single element, represented as [*str_dict1*].

4.2. TensorTable-based RA operators

This section introduces the implementations of RA operators based on the TensorTable abstraction. The supported RA operators include selection, projection, inner-join, outer-join, left-join, right-join, cross-join, group by, and aggregation.

4.2.1. Selection

The selection operator takes rows from a TensorTable that satisfy specified selection conditions.

There are three types of selection conditions:

(1) Comparison between two columns, as shown in Algorithm 1. The input is one TensorTable: $InTable$, two columns $col1$ and $col2$ to compare, and one comparison function $CompFunc$. Comparison functions include *equal*, *greater*, *less*, *greater_equal*, *less_equal*, and *not_equal*. Initially, we extract the corresponding tensors, $tensor1$ and $tensor2$, from InTable based on the specified columns. Both tensors have a shape of [m, 1], where m represents the number of rows. Subsequently, we perform element-wise comparisons between these two tensors, resulting in a mask tensor that utilizes 1 and 0 to indicate whether rows meet the selection condition. Finally, we use *nonzero* operators to get the indices that satisfy the selection condition based on the mask tensor and use the *index_select* function to extract rows from $InTable.data\_tensor$ and assign them to the *data_tensor* of the output TensorTable: $OutTable$. The *column_names*, *column_types*, and *str_dict_list* of $OutTable$ remain the same as $InTable$.

(2) Comparison between one column and a threshold. This type entails a broadcast comparison between the corresponding tensor and the specified threshold. The subsequent steps mirror those in Algorithm 1.

(3) Combination of multiple conditions. These conditions are organized using logical operators such as *and*, *or*, and *not*. We compute each condition's mask tensor and use their *intersection*, *union*, and *complement* to get the combined mask. This combined mask is then used to build the index and extract rows.

Algorithm 1 Selection

```
Input: InTable: Input TensorTable; col1, col2: Two columns to compare;
    CompFunc: Comparison function.
Output: OutTable: Output TensorTable.
1: tensor1 ← get_tensor(InTable, col1)
2: tensor2 ← get_tensor(InTable, col2)
3: mask ← CompFunc(tensor1, tensor2)
4: index ← nonzero(mask)
5: OutTable.data_tensor ← index_select(InTable.data_tensor, dim=0, index)
6: OutTable.column_names ← InTable.column_names
7: OutTable.column_types ← InTable.column_types
8: OutTable.str_dict_list ← InTable.str_dict_list
```

4.2.2. Projection

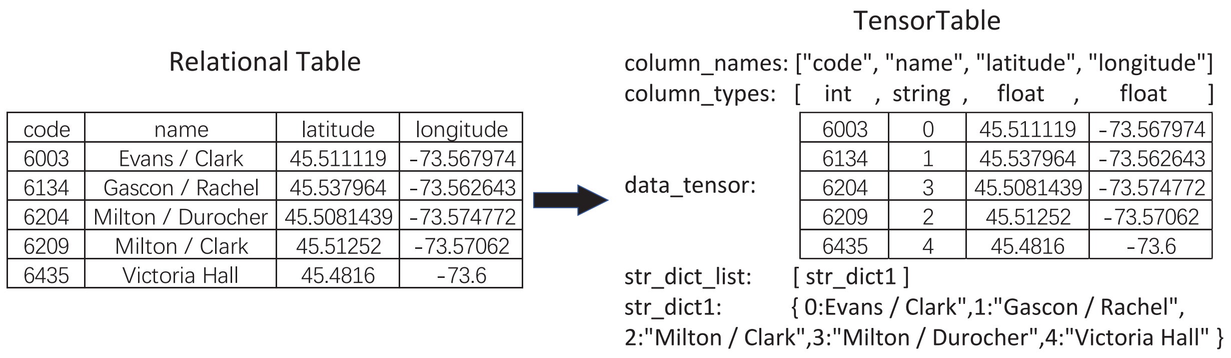
The projection operator takes columns from the input TensorTable: $InTable$ based on the specified *name_list*, which contains the selected column names, as shown in Algorithm 2. First, we parse the *name_list* to obtain the list of corresponding indices. Then we use the *index_select* function to select columns according to these indices and assign them to the *data_tensor* of the output TensorTable: $OutTable$. The *column_names* of $OutTable$ is set as *name_list*, while the *column_types* and *str_dict_list* of $OutTable$ are the subsets of $InTable$ based on *name_list*.

Algorithm 2 Projection

```
Input: InTable: Input TensorTable; name_list: Column names.
Output: OutTable: Output TensorTable.
1: index ← parse(name_list)
2: OutTable.data_tensor ← index_select(InTable.data_tensor, dim=1, index)
3: OutTable.column_names ← name_list
4: OutTable.column_types ← subset(InTable.column_types, name_list)
5: OutTable.str_dict_list ← subset(InTable.str_dict_list, name_list)
```

4.2.3. Join

The join operator combines columns from two or more input TensorTables, producing a new output TensorTable. We take inner-join as an example, as shown in Algorithm 3. The input is two TensorTables:


Relational Table
TensorTable
column_names: ["code", "name", "latitude", "longitude"]
column_types: [int , string , float , float]
data_tensor:
str_dict_list: [str_dict1]
str_dict1: { 0:Evans / Clark",1:"Gascon / Rachel", 2:"Milton / Clark",3:"Milton / Durocher",4:"Victoria Hall" }

code	name	latitude	longitude
6003	Evans / Clark	45.511119	-73.567974
6134	Gascon / Rachel	45.537964	-73.562643
6204	Milton / Durocher	45.5081439	-73.574772
6209	Milton / Clark	45.51252	-73.57062
6435	Victoria Hall	45.4816	-73.6

6003	0	45.511119	-73.567974
6134	1	45.537964	-73.562643
6204	3	45.5081439	-73.574772
6209	2	45.51252	-73.57062
6435	4	45.4816	-73.6

Fig. 5. TensorTable stores a relational table in a tensor format, mapping non-numeric values to numeric ones. For a relational table with m rows and n columns, the corresponding TensorTable consists of two string lists *column_names* and *column_types*, used to store the column names and data types, a m×n tensor *data_tensor* to store data, and a list of auxiliary dictionaries, *str_dict_list*, used to preserve the mapping relationships.

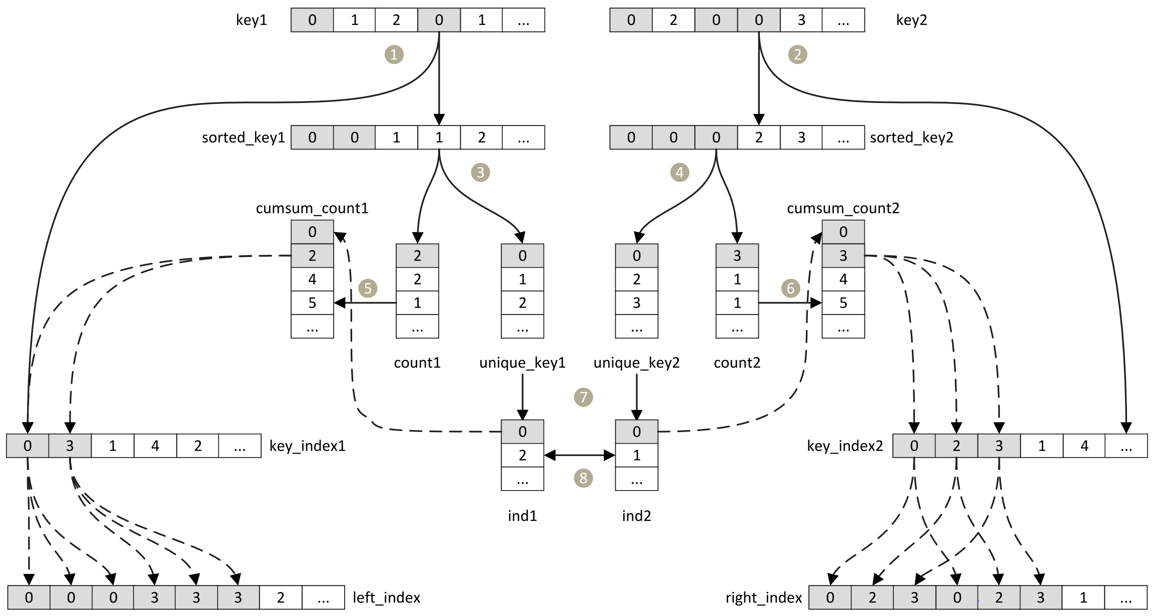


Fig. 6. The illustration of inner_join_index algorithm. Full lines represent operators, dotted lines represent indices.

Algorithm 3 Inner Join

```
Input: T1, T2: Input TensorTables; key1, key2: Join keys.
Output: OutTable: Output TensorTable.
1: key_tensor1 ← get_tensor(T1, key1)
2: key_tensor2 ← get_tensor(T2, key2)
3: left_index, right_index ← inner_join_index(key_tensor1, key_tensor2)
4: left_tensor ← index_select(T1.data_tensor, dim=0, left_index)
5: right_tensor ← index_select(T2.data_tensor, dim=0, right_index)
6: OutTable.data_tensor ← concatenate(left_tensor, right_tensor)
7: OutTable.column_names ← T1.column_names ∪ T2.column_names
8: OutTable.column_types ← T1.column_types ∪ T2.column_types
9: OutTable.str_dict_list ← T1.str_dict_list ∪ T2.str_dict_list
```

$T1$ and $T2$, and two keys to join: $key1$ from $T1$ and $key2$ from $T2$. The output TensorTable is $OutTable$. First, we extract key tensors from two TensorTables. Then we use the *inner_join_index* function to parse key tensors and return left and right indexes, which are used to select rows from two input TensorTables, as defined in Algorithm 4.

Next, we provide a detailed explanation of the *inner_join_index* algorithm in detail, as depicted in Fig. 6. This algorithm takes two key tensors as inputs and produces two indices used for selecting rows from two TensorTables. The process begins with the utilization of the *stable_sort* function to sort the two keys and get the sorted keys $sorted\_key1$ and $sorted\_key2$, and their respective indices key_index1 and key_index2, which represent the positions of elements in the original keys (lines 1–2 in Algorithm 4 and ❶❷ in Fig. 6). Subsequently, the *unique* function is applied to obtain the unique keys $unique\_key1$ and $unique\_key2$, along with their corresponding counts $count1$ and $count2$ (lines 3–4 and ❸❹). Following this, we employ the *cumsum* function to calculate the cumulative sum of counts, marked as $cumsum\_count1$ and $cumsum\_count2$ (lines 5–6 and ❺❻). We add 0 as the first element to both $cumsum\_count1$ and $cumsum\_count2$. Moving forward, we compute the intersection, denoted as $inter$, along with corresponding indices, $ind1$ and $ind2$, of the unique keys(lines 7 and ❼).

Finally, we iterate through $ind1$ and $ind2$ to construct $left\_index$ and $right\_index$ (lines 8–17 and ❽). We utilize $cumsum\_count1$ and $cumsum\_count2$ as indices to get elements from key_index1 and key_index2 and assign their values to $left\_index$ and $right\_index$ appropriately. Specifically, We take $cumsum\_count1[ind1[i]]$ and $cumsum\_count1[ind1[i]+1]$ as the index to get elements from key_index1, that is $key\_index1[cumsum\_count1[ind\text{-}1[i]], cumsum\_count1[ind1[i]+1]]$, marked as $left\_tmp$. We then repeat each element in $left\_tmp$ for $count2[ind1[i]]$ times and append them to $left\_index$. Similarly, we take $cumsum\_count2[ind2[i]]$ and $cumsum\_count2[ind2[i]+1]$ as the index to get elements from key_index2, that is $key\_index2[cumsum\_count2[ind2[i]], cumsum\_count2[ind\text{-}2[i]+1]]$, marked as $right\_tmp$. We repeat $right\_tmp$ for $count1[ind2[i]]$ times, and append them to $right\_index$. An example

Algorithm 4 inner_join_index

```
Input: key_tensor1, key_tensor2: Two keys in tensor formats.
Output: left_index, right_index: The indices to select rows from two TensorTables.
 1: sorted_key1, key_index1 ← stable_sort(key_tensor1)
 2: sorted_key2, key_index2 ← stable_sort(key_tensor2)
 3: unique_key1, count1 ← unique(sorted_key1, return_counts=True)
 4: unique_key2, count2 ← unique(sorted_key2, return_counts=True)
 5: cumsum_count1 ← cumsum(count1)
 6: cumsum_count2 ← cumsum(count2)
 7: inter, ind1, ind2 ← intersection(unique_key1, unique_key2)
 8: left_index ← Empty list
 9: right_index ← Empty list
10: for i ← 0 to len(ind1) do
11:     left_tmp ← key_index1[cumsum_count1[ind1[i]], cumsum_count1[ind1[i]+1]]
12:     Repeat each element in left_tmp for count2[ind1[i]] times
13:     Append left_tmp to left_index
14:     right_tmp ← key_index2[cumsum_count2[ind2[i]], cumsum_count2[ind2[i]+1]]
15:     Repeat right_tmp for count1[ind2[i]] times
16:     Append right_tmp to right_index
17: end for
```

is illustrated in Fig. 6; the gray part shows where $key1$ and $key2$ are all 0, which means the variable i used in line 10 and step ⑧ is 0. $Cumsum\_count1[0]$ and $cumsum\_count1[1]$ are 0 and 2, so we take the first two elements from key_index1, that is [0, 3]. Count2[0] is 3, so each element in [0, 3] is repeated three times, yielding [0, 0, 0, 3, 3, 3], which is appended to $left\_index$. $Cumsum\_count2[0]$ and $cumsum\_count2[1]$ are 0 and 3, so we take the first three elements from key_index2, that is [0, 2, 3]. Count1[0] is 2, so [0, 2, 3] is repeated twice, yielding [0, 2, 3, 0, 2, 3], which is appended to $right\_index$.

After the *inner_join_index* function, we select rows from two TensorTables based on $left\_index$ and $right\_index$, concatenate them, and assign them to $OutTable.data\_tensor$. The *column_names*, *column_types*, and *str_dict_list* of $OutTable$ are the union of that of two input TensorTables. Outer-join, left-join, right-join, and cross-join have similar implementations, using their specific *join_index* algorithms to replace *inner_join_index*.

4.2.4. Groupby

The groupby operator groups rows based on one or more columns. The input is one TensorTable $InTable$ and a $GroupbyCol$ which marks the column names to make groups. The output is one TensorTable $OutTable$. First, we parse the $GroupbyCol$ and get the list of indices. Then we utilize the *index_select* function to select the column based on indices and assign the values to $GroupbyTensor$. Subsequently, we utilize the *unique* function to get the unique elements and generate $GroupbyIdx$, which marks the group to which each row belongs. Finally, we append $GroupbyIdx$ to *data_tensor*, append "group" to *column_names*, and append int32 to *column_types* of $OutTable$. The *str_dict_list* of $OutTable$ remains the same as that of $InTable$.

Algorithm 5 Groupby

```
Input: InTable: Input TensorTable; GroupbyCol: Groupby column name.
Output: OutTable: Output TensorTable.
1: index ← parse(GroupbyCol)
2: GroupbyTensor ← index_select(InTable, dim=1, index)
3: UniqueElement, GroupbyIdx ← unique(GroupbyTensor, return_inverse=True)
4: OutTable.data_tensor ← concatenate(GroupbyIdx, InTable.data_tensor)
5: OutTable.column_names ← InTable.column_names
6: OutTable.column_names ← ["group"] ∪ InTable.column_names
7: OutTable.column_types ← [int32] ∪ InTable.column_types
8: OutTable.str_dict_list ← InTable.str_dict_list
```

4.2.5. Aggregation

The aggregation operator collects one or more columns and returns their aggregated values. The input is one TensorTable $InTable$, one aggregation function $AggFunc$, and the *name_list* which marks the column names. The output is one TensorTable $OutTable$. Aggregation functions include *count*, *sum*, *avg*, *min*, *max*, etc. First, we parse the *name_list* and get the list of indices. Then we use the *index_select* function to select the columns based on these indices and assign them to $AggTensor$. Finally, we make aggregation on $AggTensor$ and assign the results to $OutTable.data\_tensor$. The *column_names* of $OutTable$ is set as *name_list*. The *column_types* and *str_dict_list* of $OutTable$ are the subsets of $InTable$ according to *name_list*.

Algorithm 6 Aggregation

```
Input: InTable: Input TensorTable; AggFunc: Aggregation function; name_list: Column names.
Output: OutTable: Output TensorTable.
1: index ← parse(name_list)
2: AggTensor ← index_select(InTable.data_tensor, dim=1, index)
3: OutTable.data_tensor ← AggFunc(AggTensor)
4: OutTable.column_names ← name_list
5: OutTable.column_types ← subset(InTable.column_types, name_list)
6: OutTable.str_dict_list ← subset(InTable.str_dict_list, name_list)
```

4.3. The mixed RA and LA pipeline implementation

This section presents the implementation of mixed RA and LA pipelines.

4.3.1. DAG IR

First, we introduce the Directed Acyclic Graph (DAG) Intermediate Representation (IR), which serves as the representation for mixed pipelines. The DAG IR comprises a list of operators, variables, and utility functions used to build pipelines and execute them. The variables are all TensorTables, as defined in Section 4.1. These TensorTables form the fundamental data units within the representation. The operators take TensorTables as both input and output, including both linear and relational operators. Within the DAG IR, nodes represent operators, while edges represent variables and mark the data dependencies between operators. In this way, this unified abstraction allows for the seamless representation of mixed RA and LA pipelines.

4.3.2. Parser

Parser traverses the source codes and transforms them into DAG IRs. Operators are initialized as nodes within the DAG IR, marking their input and output variables, along with an implementation instance used to lower and execute operators. Variables are established as edges within the DAG IR, facilitating connections between nodes. This comprehensive process results in the construction of a corresponding DAG IR, which encapsulates the entire representation once all source code statements have been traversed.

4.3.3. Optimizer

Optimizer performs a series of functionally equivalent transformations for DAG IRs aimed at achieving optimizations. These optimizations primarily fall into two categories: *operator swap* and *operator fusion*.

Operator swap involves altering the order of operators within the DAG, which can reduce computation and create more opportunities for *operator fusion*. RA operators such as *selection* and *projection* can be swapped with other RA and LA operators without affecting the result. This optimization is similar to *pushdown* used in the RA systems [32, 33], but is expended to LA. For instance, consider a scenario where users calculate the reciprocal of one column and then make a selection. By rearranging the sequence to perform selection first and then the reciprocal operation, we can reduce the computation.

Operator fusion fuses two operators to reduce memory footprint and computation. This optimization is similar to that in the LA systems [34,35], but is expended to RA operators. We can effectively harness PyTorch to fuse linear operators and, simultaneously, apply fusion techniques to RA operators based on our TensorTable-based implementations. For instance, multiple selections or projections on a single TensorTable can be fused into a single operation. Similarly, multiple join operations can be consolidated into a single operation. For example, if users initially join A and B, and then join the result with C, we can fuse these operations into a single join. This involves calculating the indices for A, B, and C, as detailed in Section 4.2.3. Subsequently, we select rows from A, B, and C and concatenate them without generating intermediate tables.

4.3.4. Code generator

Code Generator converts those optimized DAGs to TensorTable-based operators based on the implementation instances of operators in DAG IRs. When dealing with RA operators, we adhere to the implementations outlined in Section 4.2 and translate them into a list of tensor operations as well as some simple auxiliary functions to handle TensorTables. In the case of LA operators, we translate them into PyTorch LA operators to handle the *data_tensor* in TensorTables and do not deal with other proprieties.

4.3.5. Executor

We construct the Executor on top of PyTorch, orchestrating the execution flow through a series of carefully sequenced program calls. In the case of RA operators, we employ PyTorch to execute the TensorTable-based operators. Meanwhile, auxiliary functions are executed using native Python. For LA operators, we call PyTorch to handle TensorTables and keep other proprieties just the same. We use an interpreted mode in PyTorch to reduce compilation time.

5. Evaluation

5.1. Experimental setup

We deploy a server node equipped with two Xeon E5-2620 V3 (Haswell) CPUs and 64 GB memory to conduct the experiments. Each CPU contains six physical cores. The operating system is Ubuntu 16.04, and the other software includes PyTorch 1.13, pandas 1.3.5, Spark 3.3.0, MonetDB 11.39.17, AIDA [18], and RMA [29].

5.2. The RA operator performance

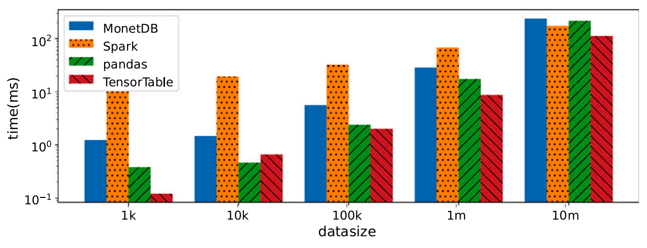
This section evaluates RA operators. We use the BIXI dataset [31] and test the results on 1k, 10k, 100k, 1 m, and 10 m rows. We compare our work against MonetDB, Spark, and pandas.

Fig. 7(a) shows the performance of the selection operator. TensorTable outperforms the other frameworks on both small and large data sizes for two reasons. First, TensorTable stores data in a tensor format, allowing for faster data access compared to row-oriented or column-oriented data structures. Additionally, TensorTable utilizes element-wise comparison based on vector instructions to accelerate.

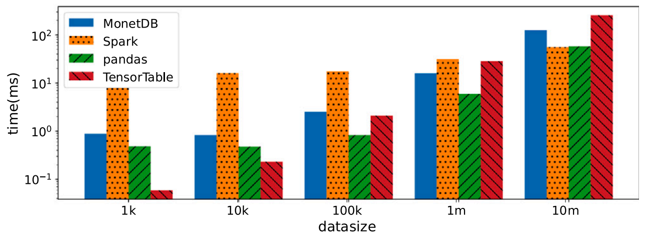
Fig. 7(b) presents the performance of the projection operator. TensorTable achieves noticeable speedup on small datasets due to better data locality and faster data access. However, TensorTable's performance diminishes on larger datasets due to the use of the *index_select* function, which reassigns the tensor data structure to ensure data contiguity and causes redundant computation. Utilizing the *index_select* function is essential, as it can maintain a good data locality and guarantee the performance of other operators, especially linear operators.

Fig. 7(c) displays the performance of the inner join operator. MonetDB and Spark use hash-based join, while pandas and TensorTable use sort-based join. TensorTable continues to outperform the others on small datasets. However, hash-based joins have lower complexity than sort-based joins, giving Spark and MonetDB an advantage as the dataset size increases. We are actively exploring the integration of hash-based joins into TensorTable without compromising the performance of other tensor computations.

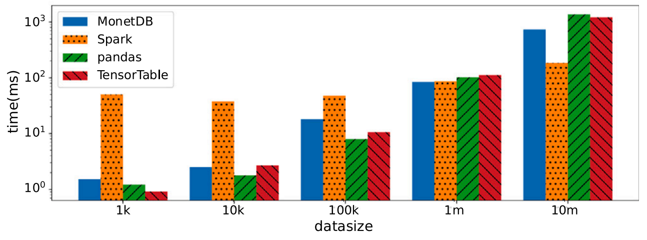
Fig. 7(d) demonstrates the performance of the groupby operator, which exhibits similarities to the inner join operator. TensorTable excels on small datasets but lags behind on larger datasets due to limitations of the sort-based implementation. We are actively investigating the implementation of hash-based groupby operator in TensorTable.



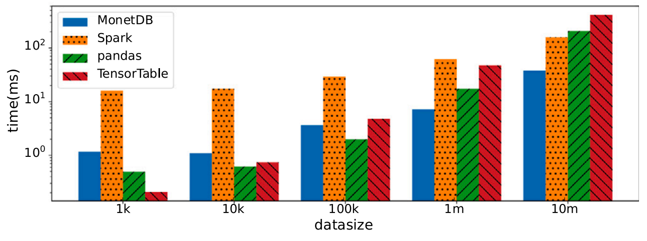
(a) Selection



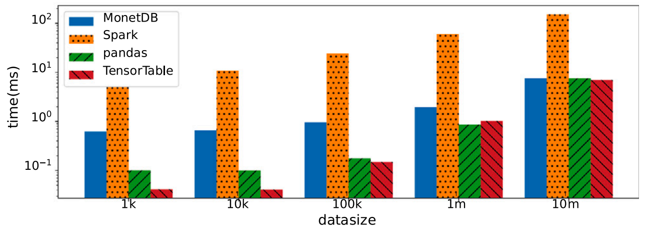
(b) Projection



(c) Inner Join



(d) Groupby



(e) Aggregation

Fig. 7. The execution time of RA operators on TensorTable, MonetDB, Spark, and pandas over different data sizes.

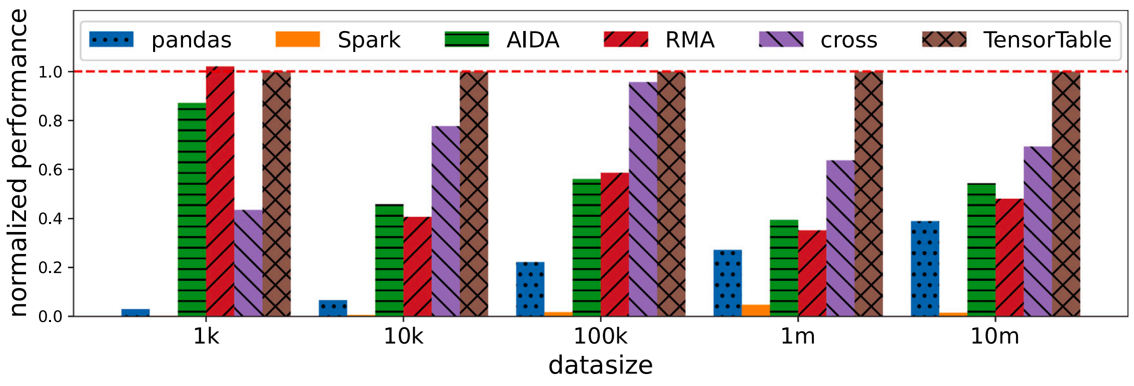


Fig. 8. The normalized performance of the distance-duration linear regression pipeline on different data sizes. Using the performance of TensorTable to normalize other frameworks, a smaller number has worse performance. "Cross" represents the cross-framework implementation using pandas for RA operators and PyTorch for LA operators.

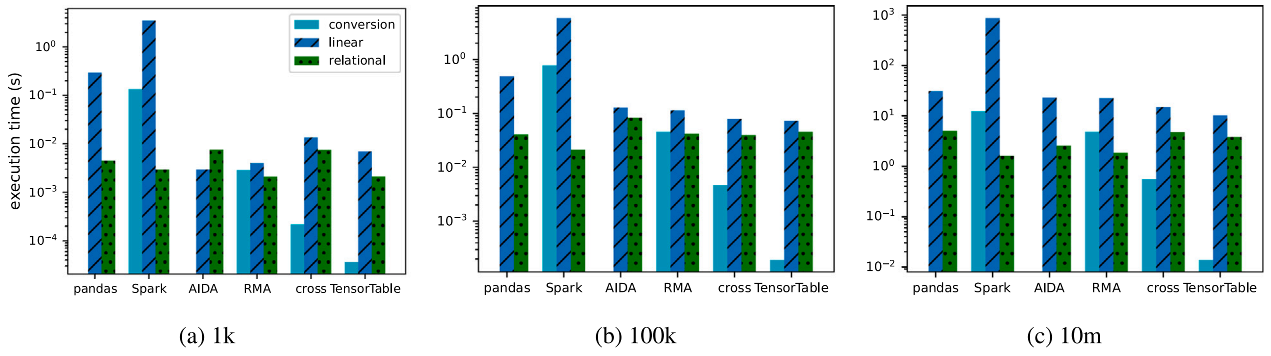


Fig. 9. The execution time breakdown of the distance-duration linear regression pipeline on different data sizes. It consists of three primary parts: data conversion, LA operators, and RA operators.

Fig. 7(e) illustrates the performance of the aggregation operator. This operator relies more on linear algebra computations than other relational operators, which benefits TensorTable on both small and large datasets.

In summary, TensorTable achieves superior performance on selection and aggregation operators across all data sizes. For projection, inner join, and group by operators, TensorTable performs better on small datasets while worse on large datasets. TensorTable achieves competitive performance on RA operators compared to RA and general-purpose systems.

5.3. *Mixed pipeline performance*

This section evaluates two mixed pipelines consisting of RA and LA operators.

5.3.1. *Distance-duration linear regression*

The first case derives from a public bicycle sharing system [18] that involves linear regression between distance and duration, utilizing the BIXI dataset [31]. There are two tables: *trip* contains start stations, end stations, and duration; *station* contains station codes, names, and coordinates. This pipeline comprises five steps: (1) Select rows from *trip* where the start station does not equal the end station. (2) Join *trip* with *station* to retrieve the coordinates of start and end stations. (3) Calculate the distance. (4) Train a linear regression model between distance and duration. (5) Test the model on the test dataset. Steps (1)(2) involves RA operators, and steps (3)-(5) involves LA operators. We do not use any built-in linear regression algorithms to avoid bias from these algorithms' implementation differences. We use those basic LA operators, such as matrix multiplication, to implement linear regression, guaranteeing consistency in computation for different frameworks.

We compare TensorTable with pandas, Spark, AIDA, RMA, and cross-framework implementation, which uses pandas for RA operators and PyTorch for LA operators. We use the performance of TensorTable to normalize the other five approaches, as illustrated in Fig. 8. Note that a value smaller than 1 indicates a worse performance compared to TensorTable and the smaller the value, the worse the performance. TensorTable achieves a 2.57x-32.33x speedup compared with pandas, a 20.77x-390.55x speedup compared with Spark, a 1.15x-2.53x speedup compared with AIDA, and a 1.04x-2.29x speedup compared with cross-framework implementation. Although TensorTable lags behind RMA by 2% when dealing with small datasets, it outperforms RMA as the dataset size grows, with a speedup ranging from 1.71x to 2.84x.

Fig. 9 provides a detailed breakdown of the results. The execution time of this mixed pipeline is divided into three primary components: data conversion, LA operators, and RA operators. Our work converts relational tables into TensorTables during initialization and does not need other data conversion in later computation, whose data conversion time is not apparent. Pandas handles DataFrames for both LA and RA operators without explicit data conversion, resulting in nearly zero conversion time. TensorTable achieves a 3.04x-41.71x speedup for LA operators and a 1.11x-2.09x speedup for RA operators compared with pandas. Spark implements RA operators using DataFrames and LA operators using matrices, necessitating frequent data conversion. While Spark excels in RA operators for sizable datasets, it lags in LA operators and involves more data conversion, resulting in the worst performance among the six implementations. TensorTable achieves a 41.44x-491.96x speedup for LA operators and an 868.25x-4546.49x speedup for data conversion compared with Spark. AIDA handles TabularData for both LA and RA operators without explicit data conversion, resulting in almost negligible conversion time. RMA handles RA operators and simple LA operators like addition based on binary association tables and executes complex LA operators such as matrix multiplication by calling external libraries like MKL [36]. This incurs notable data conversion costs, particularly with complex LA operators. AIDA

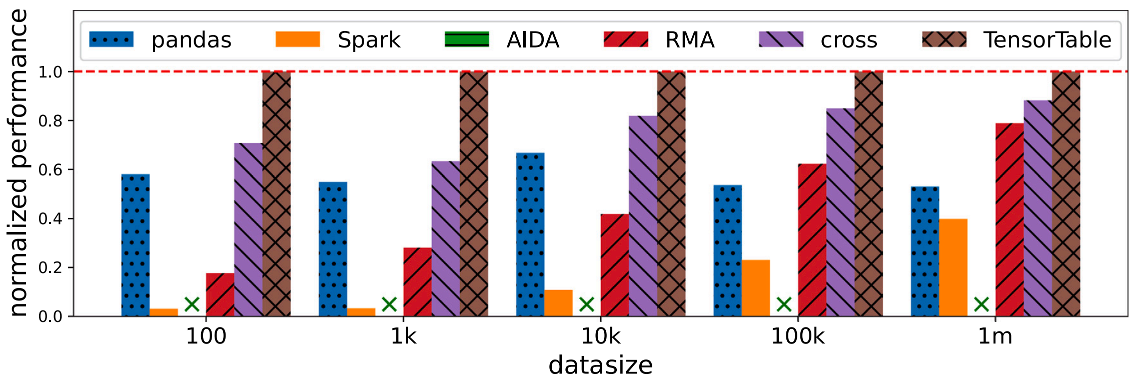


Fig. 10. The normalized performance of the conference covariance pipeline on different data sizes. Using the performance of TensorTable to normalize other frameworks, a smaller number has worse performance. “Cross” represents the cross-framework implementation using pandas for RA operators and PyTorch for LA operators. “x” means unsupported.

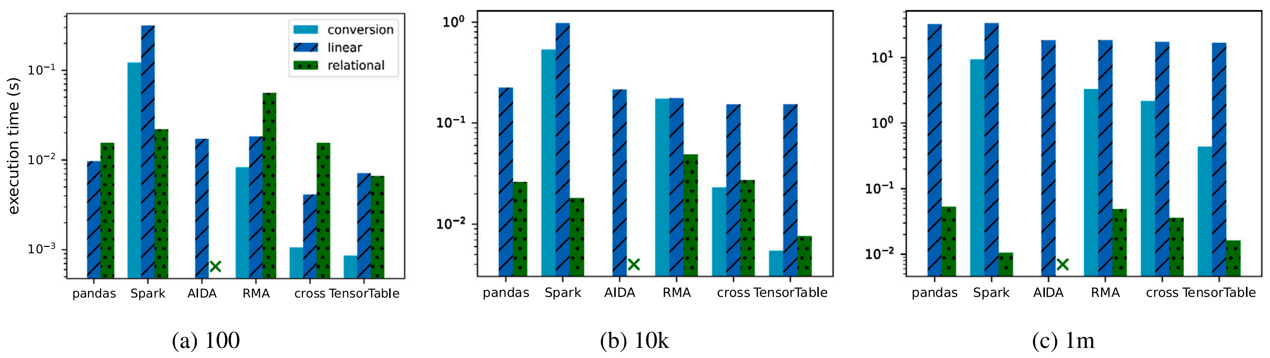


Fig. 11. The execution time breakdown of the conference covariance pipeline on different data sizes. It consists of three primary parts: data conversion, LA operators, and RA operators. “x” means unsupported.

uses numpy for LA operators and RMA uses MKL for LA operators. These two libraries provide better optimizations for small-scale LA computations over PyTorch. Therefore, AIDA and RMA outperform TensorTable and cross-framework implementation for LA operators over smaller datasets. However, as the dataset grows, PyTorch demonstrates superior operator-level and graph-level optimizations, ensuring TensorTable and cross-framework implementation's superiority in LA operators for larger datasets. TensorTable achieves a 1.76x-4.12x speedup for LA operators and a 1.82x-3.54x speedup for RA operators compared with AIDA. TensorTable achieves a 1.58x-3.72x speedup for LA operators and a 78.22x-344.62x speedup for data conversion compared with RMA. While RMA exhibits slightly better performance on RA operators compared with TensorTable, its data conversion and LA operators compromise its end-to-end performance. The LA performance of TensorTable and cross-framework implementation have no significant differences. However, cross-framework implementation requires frequent data conversion between DataFrames and tensors during runtime. TensorTable achieves a 1.03x-2.61x speedup for RA operators and a 5.14x-39.43x speedup for data conversion compared with cross-framework implementation.

5.3.2. *Conferences–covariance computation*

The second case derives from an academic conference management system [29], it computes the covariance between A++ conferences and other conferences based on the number of publications per author and conference, using the DBLP dataset [37]. There are two tables: *ranking* stores conferences and their ratings, and *publication* stores the number of publications per author and conference. The pipeline involves three main steps: (1) Compute the covariance matrix on *publication*. (2) Join the result with *ranking*. (3) Select the A++ conferences. Step (1) uses LA operators, and Steps (2)(3) use RA operators.

Fig. 10 displays the normalized performance of TensorTable compared to pandas, Spark, AIDA, RMA, and cross-framework implementation using pandas for RA operators and PyTorch for LA operators. TensorTable achieves a 1.5x-1.88x speedup compared with pandas, a 2.51x-31.31x speedup compared with Spark, a 1.27x-5.63x speedup compared with RMA, and a 1.13x-1.58x speedup compared with cross-framework implementation. AIDA loses contextual information when performing LA operators on TabularData containing non-numeric columns. This results in some RA operators failing to execute after LA operators. As a result, this mixed pipeline can only execute the first half, encountering errors in the latter part for AIDA.

Fig. 11 offers a breakdown of execution time into three primary parts: data conversion, LA operators, and RA operators. TensorTable achieves a 1.36x-1.93x speedup for LA operators and a 1.97x-3.47x speedup for RA operators compared with pandas. TensorTable achieves a 2.01x-44.1x speedup, a 1.5x-3.33x speedup, and a 21.62x-249.57x speedup for LA operators, RA operators, and data conversion, respectively, compared with Spark. TensorTable achieves a 1.1x-2.42x speedup for LA operators compared with AIDA, with AIDA's RA operators encountering errors due to contextual information loss caused by preceding LA operators. Against RMA, TensorTable achieves a 1.1x-2.56x speedup for LA operators, a 3.02x-8.4x speedup for RA operators, and a 7.27x-31.66x speedup for data conversion. The LA performance of TensorTable and cross-framework implementation have no significant differences. However, TensorTable achieves a 2.22x-3.62x speedup for RA operators and a 1.23x-4.96x speedup for data conversion in contrast to cross-framework implementation. In summary, TensorTable attains optimal end-to-end performance by leveraging high-performance LA operators, minimizing data conversion overhead, and achieving competitive performance on RA operators.

6. Related work

There are four categories of system implementations to support the mixed RA and LA pipelines.

(1) Extending SQL for LA [24–26,38–42]. These works leverage the mature ecosystem of Relational Database Management Systems (RDBMS) and compile LA computations into SQL statements or RA expressions. While they can ensure good performance for RA operators, the SQL statements often limit the optimization potential for LA computations, leading to suboptimal performance. Additionally, certain LA operators, such as matrix inversion and determinant computation, cannot be implemented using this method.

(2) Using User-Defined Actions (UDAs) or User-Defined Functions (UDFs) to implement LA in RDBMS [27,28,43–45]. However, it requires significant effort and lacks flexibility. UDAs or UDFs provide technical interfaces but do not offer ready-made implementations, leaving users to create high-performance implementations themselves, which can be challenging. Furthermore, these interfaces may struggle to handle the various shapes and dimensions of LA operators effectively, and hinder the users' incremental developments, such as changing algorithms and tuning parameters.

(3) Building cross-framework applications [17–20]. These approaches utilize different systems for specific tasks, combining scientific computing systems for LA operators, and RDBMS for RA operators, and then integrating them. While this method can cover a wide range of LA and RA operators, it introduces extra costs due to data copying and transformations between frameworks and may limit cross-framework optimizations.

(4) Proposing new abstractions for both LA and RA operators. Some of them [22,23,46] propose new abstractions and build dedicated data analysis frameworks from scratch. Others [29,30] propose new abstractions and extend existing frameworks. Most of these systems are based on RDBMS, prioritizing performance for RA operators. However, they often lack optimizations for LA operators, which cover more execution time. In contrast, our work proposes a new abstraction and implements the system on the LA framework, ensuring optimal performance for LA operators while still accommodating RA operators.

7. Conclusion

This paper introduces TensorTable, a novel abstraction designed to seamlessly accommodate mixed pipelines encompassing both relational algebra (RA) and linear algebra (LA) operators. TensorTable can represent relational tables from RA, as well as vectors, matrices, and tensors from LA. we provide TensorTable-based implementations for RA operators and build a system that supports mixed LA and RA pipelines. Built on top of PyTorch, our implementation ensures comparable performance across both RA and LA operators, especially on small datasets. Besides, TensorTable achieves a 1.15x-5.63x speedup for mixed pipelines, outperforming state-of-the-art frameworks—AIDA and RMA.

CRediT authorship contribution statement

Xu Wen: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Data curation, Conceptualization.

Declaration of competing 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

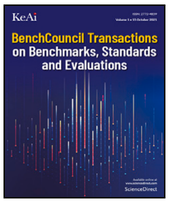
[1] Per-Åke Larson, Adrian Birka, Eric N. Hanson, Weiyun Huang, Michal Nowakiewicz, Vassilis Papadimos, Real-time analytical processing with SQL server, Proc. VLDB Endow. 8 (12) (2015) 1740–1751.
[2] Babak Yadranjiaghdam, Nathan Pool, Nasseh Tabrizi, A survey on real-time big data analytics: applications and tools, in: 2016 International Conference on Computational Science and Computational Intelligence, CSCI, IEEE, 2016, pp. 404–409.
[3] Arun Kejariwal, Sanjeev Kulkarni, Karthik Ramasamy, Real time analytics: algorithms and systems, 2017, arXiv preprint arXiv:1708.02621.
[4] Erum Mehmood, Tayyaba Anees, Challenges and solutions for processing real-time big data stream: a systematic literature review, IEEE Access 8 (2020) 119123–119143.
[5] Shivani S. Kale, Preeti S. Patil, A machine learning approach to predict crop yield and success rate, in: 2019 IEEE Pune Section International Conference (PuneCon), IEEE, 2019, pp. 1–5.
[6] Marcus D. Bloice, Andreas Holzinger, A tutorial on machine learning and data science tools with python, in: Machine Learning for Health Informatics: State-of-the-Art and Future Challenges, Springer, 2016, pp. 435–480.
[7] M. Vagizov, A. Potapov, K. Konzhgoladze, S. Stepanov, I. Martyn, Prepare and analyze taxation data using the python pandas library, in: IOP Conference Series: Earth and Environmental Science, Vol. 876, (1) IOP Publishing, 2021, 012078.
[8] Ricardo Silva Peres, Andre Dionisio Rocha, Paulo Leitao, Jose Barata, IDARTS–towards intelligent data analysis and real-time supervision for industry 4.0, Comput. Ind. 101 (2018) 138–146.
[9] Chunquan Li, Yaqiong Chen, Yuling Shang, A review of industrial big data for decision making in intelligent manufacturing, Eng. Sci. Technol., Int. J. 29 (2022) 101021.
[10] Dharmitha Ajerla, Sazia Mahfuz, Farhana Zulkernine, A real-time patient monitoring framework for fall detection, Wirel. Commun. Mob. Comput. 2019 (2019) 1–13.
[11] Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, Martin L. Kersten, Monetdb: Two decades of research in column-oriented database architectures, IEEE Data Eng. Bull. 35 (2012) 40–45.
[12] Paul DuBois, MySQL, Pearson Education, 2008.
[13] Bruce Momjian, PostgreSQL: Introduction and Concepts, Vol. 192, Addison-Wesley New York, 2001.
[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, Travis E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362.
[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: A system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI '16, USENIX Association, USA, 2016, pp. 265–283.
[17] David Kernert, Frank Köhler, Wolfgang Lehner, SLACID-sparse linear algebra in a column-oriented in-memory database system, in: Proceedings of the 26th International Conference on Scientific and Statistical Database Management, 2014, pp. 1–12.
[18] Joseph Vinish D'silva, Florestan De Moor, Bettina Kemme, AIDA: Abstraction for advanced in-database analytics, Proc. VLDB Endow. 11 (11) (2018) 1400–1413.
[19] Stefan Hagedorn, Steffen Kläbe, Kai-Uwe Sattler, Putting pandas in a box, in: CIDR, 2021.
[20] Alekh Jindal, K. Venkatesh Emani, Maureen Daum, Olga Poppe, Brandon Haynes, Anna Pavlenko, Ayushi Gupta, Karthik Ramachandra, Carlo Curino, Andreas Mueller, et al., Magpie: Python at speed and scale using cloud backends, in: CIDR, 2021.
[21] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, Spark: Cluster computing with working sets, in: 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10), 2010.
[22] Michael Armbrust, Reynolds S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, et al., Spark sql: Relational data processing in spark, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1383–1394.

[23] Wes McKinney, et al., Pandas: a foundational python library for data analysis and statistics, Python High Perform. Sci. Comput. 14 (9) (2011) 1–9.
[24] Francesco Del Buono, Matteo Paganelli, Paolo Sottovia, Matteo Interlandi, Francesco Guerra, Transforming ML predictive pipelines into SQL with MASQ, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2696–2700.
[25] Shangyu Luo, Zekai J. Gao, Michael N. Gubanov, Luis Leopoldo Perez, Christopher M. Jermaine, Scalable linear algebra on a relational database system, IEEE Trans. Knowl. Data Eng. 31 (7) (2019) 1224–1238.
[26] Ying Zhang, Martin Kersten, Stefan Manegold, SciQL: Array data processing inside an RDBMS, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 1049–1052.
[27] Konstantinos Karanasos, Matteo Interlandi, Doris Xin, Fotis Psallidas, Rathijit Sen, Kwanghyun Park, Ivan Popivanov, Supun Nakandal, Subru Krishnan, Markus Weimer, et al., Extending relational query processing with ML inference, 2019, arXiv preprint arXiv:1911.00231.
[28] Carlos Ordonez, Building statistical models and scoring with UDFs, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007, pp. 1005–1016.
[29] Oksana Dolmatova, Nikolaus Augsten, Michael H. Böhlen, A relational matrix algebra and its implementation in a column store, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 2573–2587.
[30] Dylan Hutchison, Bill Howe, Dan Suciu, Laradb: A minimalist kernel for linear and relational algebra computation, in: Proceedings of the 4th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and beyond, 2017, pp. 1–10.
[31] Ahmadreza Faghih-Imani, Naveen Eluru, Ahmed M. El-Geneidy, Michael Rabbat, Usama Haq, How land-use and urban form impact bicycle flows: evidence from the bicycle-sharing system (BIXI) in Montreal, J. Transp. Geogr. 41 (2014) 306–314.
[32] Surajit Chaudhuri, An overview of query optimization in relational systems, in: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '98, ACM, New York, NY, USA, 1998, pp. 34–43.
[33] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts, third ed., McGraw-Hill, New York, 2010.
[34] Tianqi Chen, Lianmin Zheng, Eddie Yan, Ziheng Jiang, Thierry Moreau, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy, Learning to optimize tensor programs, Adv. Neural Inf. Process. Syst. 31 (2018).
[35] Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, Alex Aiken, TASO: optimizing deep learning computation with automatic generation of graph substitutions, in: Proceedings of the 27th ACM Symposium on Operating Systems Principles, 2019, pp. 47–62.
[36] Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, Yajuan Wang, Endong Wang, Qing Zhang, Bo Shen, et al., Intel math kernel library, in: High-Performance Computing on the Intel® Xeon Phi™: how to Fully Exploit MIC Architectures, Springer, 2014, pp. 167–188.
[37] University of Trier, DBLP computer science bibliography, 2022, https://dblp.uni-trier.de/.
[38] Vladimir Kotlyar, Keshav Pingali, Paul Stodghill, A relational approach to the compilation of sparse matrix programs, in: European Conference on Parallel Processing, Springer, 1997, pp. 318–327.
[39] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, N. Widmann, The multidimensional database system RasDaMan, SIGMOD Rec. 27 (2) (1998) 575–577.
[40] Dimitrije Jankov, Shangyu Luo, Binhang Yuan, Zhuhua Cai, Jia Zou, Chris Jermaine, Zekai J. Gao, Declarative recursive computation on an RDBMS: Or, why you should use a database for distributed machine learning, Proc. VLDB Endow. 12 (7) (2019) 822–835.
[41] Umar Syed, Sergei Vassilvitskii, SQML: large-scale in-database machine learning with pure SQL, in: Proceedings of the 2017 Symposium on Cloud Computing, 2017, pp. 659–659.
[42] Yisu Remy Wang, Shana Hutchison, Jonathan Leang, Bill Howe, Dan Suciu, SPORES: sum-product optimization via relational equality saturation for large scale linear algebra, 2020, arXiv preprint arXiv:2002.07951.
[43] Xixuan Feng, Arun Kumar, Benjamin Recht, Christopher Ré, Towards a unified architecture for in-RDBMS analytics, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 325–336.
[44] Joe Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, et al., The MADlib analytics library or MAD skills, the SQL, 2012, arXiv preprint arXiv:1208.4165.
[45] Shreya Prasad, Arash Fard, Vishrut Gupta, Jorge Martinez, Jeff LeFevre, Vincent Xu, Meichun Hsu, Indrajit Roy, Large-scale predictive analytics in vertica: Fast data transfer, distributed model creation, and in-database prediction, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 1657–1668.
[46] R. Core Team, et al., R: A language and environment for statistical computing, 2013.

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/



Research article

Evaluatology: The science and engineering of evaluation

Jianfeng Zhan [a,b,c,*], Lei Wang [b,a,c], Wanling Gao [b,a,c], Hongxiao Li [b,c], Chenxi Wang [b,c], Yunyou Huang [d,a], Yatao Li [f], Zhengxin Yang [b,c], Guoxin Kang [b,a,c], Chunjie Luo [b,a,c], Hainan Ye [g,b,c], Shaopeng Dai [c], Zhifei Zhang [e,a]

[a] The International Open Benchmark Council, Delaware, USA
[b] ICT, Chinese Academy of Sciences, Beijing, China
[c] University of Chinese Academy of Sciences, Beijing, China
[d] Guangxi Normal University, Guilin, Guangxi, China
[e] Capital Medical University, Beijing, China
[f] Microsoft Research Asia, Beijing, China
[g] Hong Kong International Evaluation and Benchmark Research, Hong Kong, China

ARTICLE INFO

Keywords:
Evaluation
Benchmark
Scale
Index
Evaluation condition
Evaluation model
Evaluation system
Evaluation standard
Equivalent evaluation condition
Least equivalent evaluation condition
Evaluatology
Benchmarkology

ABSTRACT

Evaluation is a crucial aspect of human existence and plays a vital role in each field. However, it is often approached in an empirical and ad-hoc manner, lacking consensus on universal concepts, terminologies, theories, and methodologies. This lack of agreement has significant consequences. This article aims to formally introduce the discipline of evaluatology, which encompasses the science and engineering of evaluation. We propose a universal framework for evaluation, encompassing concepts, terminologies, theories, and methodologies that can be applied across various disciplines, if not all disciplines.

Our research reveals that the essence of evaluation lies in conducting experiments that intentionally apply a well-defined evaluation condition to individuals or systems under scrutiny, which we refer to as the *subjects*. This process allows for the creation of an evaluation system or model. By measuring and/or testing this evaluation system or model, we can infer the impact of different subjects. Derived from the essence of evaluation, we propose five axioms focusing on key aspects of evaluation outcomes as the foundational evaluation theory. These axioms serve as the bedrock upon which we build universal evaluation theories and methodologies. When evaluating a single subject, it is crucial to create evaluation conditions with different levels of equivalency. By applying these conditions to diverse subjects, we can establish reference evaluation models. These models allow us to alter a single independent variable at a time while keeping all other variables as controls. When evaluating complex scenarios, the key lies in establishing a series of evaluation models that maintain transitivity. Building upon the science of evaluation, we propose a formal definition of a benchmark as a simplified and sampled evaluation condition that guarantees different levels of equivalency. This concept serves as the cornerstone for a universal benchmark-based engineering approach to evaluation across various disciplines, which we refer to as benchmarkology.

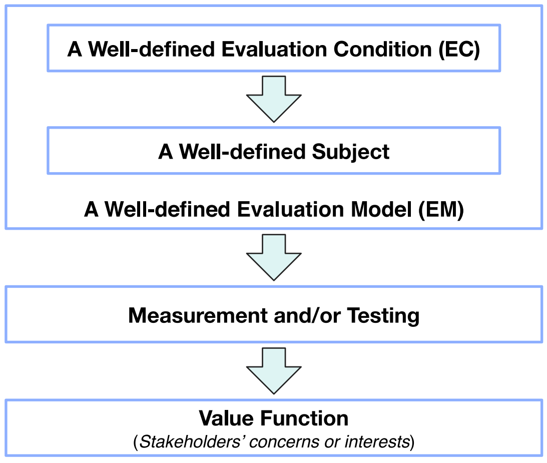
1. Introduction

Evaluation, a fundamental and significant undertaking in human existence, possesses a multifaceted nature. It spans a wide spectrum of domains, encompassing the assessment of computer performance, the evaluation of societal interventions to determine their efficacy [1], the ranking of educational institutions, and even the appraisal of political leaders through electoral processes. As a result, evaluation assumes a pivotal role that permeates every discipline. Nevertheless, it is pertinent to recognize that evaluation practices often adopt ad-hoc and empirical approaches, displaying inherent variations among various disciplines.

Collectively, evaluations within diverse disciplines lack universal concepts, terminologies, theories, and methodologies. In the disciplines of computer science, social sciences, and psychology, the communities develop different methodologies that design experiments to deliberately impose conditions on individuals or systems under scrutiny, which we refer to as the *subject*, to measure and analyze their responses [2]. In the field of computer science, a *benchmark* is utilized as a tool

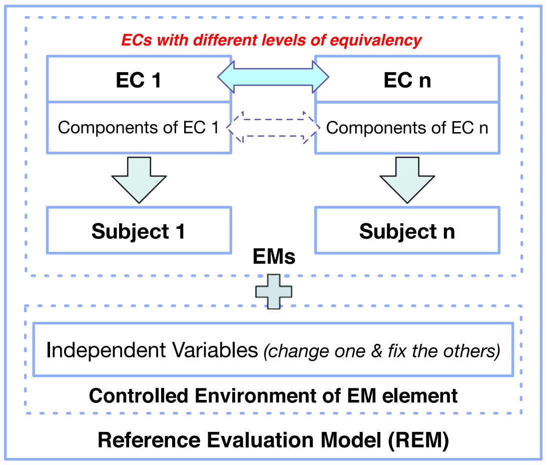
* Corresponding author.
E-mail address: jianfengzhan.benchcouncil@gmail.com (J. Zhan).
URL: https://www.zhanjianfeng.org (J. Zhan).

https://doi.org/10.1016/j.tbench.2024.100162
Received 19 March 2024; Accepted 22 April 2024
Available online 30 April 2024



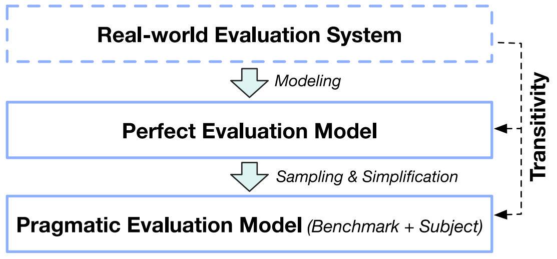
A Well-defined Evaluation Condition (EC)
A Well-defined Subject
A Well-defined Evaluation Model (EM)
Measurement and/or Testing
Value Function
(Stakeholders' concerns or interests)

(a) The essence of evaluation.



ECs with different levels of equivalency
EC 1
EC n
Components of EC 1
Components of EC n
Subject 1
Subject n
EMs
Independent Variables (change one & fix the others)
Controlled Environment of EM element
Reference Evaluation Model (REM)

(b) The basic methodology of evaluating a single subject.



Real-world Evaluation System
Modeling
Perfect Evaluation Model
Sampling & Simplification
Pragmatic Evaluation Model (Benchmark + Subject)
Transitivity

(c) The evaluation methodology addressing complexities that arise in more intricate scenarios.

Fig. 1. The universal concepts, theories, and methodologies in evaluatology.

and methodology [3–6] to evaluate the effectiveness and efficiency of system designs and implementations. In the realm of social sciences, evaluation assumes the application of social research methodologies to systematically investigate the effectiveness and efficiency of intervention programs aimed at enhancing societal conditions, as defined by Rossi et al. [1]. Within the psychology domain, social and personality psychologists often employ *scales* such as psychological inventories, tests, or questionnaires [7] to quantify psychometric variables [7].

Conversely, evaluations in the business, finance, and education domains take different observational study methodologies [2]. The field of business embraces the concept of *benchmarking* as a means to identify exemplary practices that can propel companies towards superior performance [8]. In the realms of finance and education, evaluation often utilizes a tool named *index*. Widely employed to gauge the overall performance of the system, the index is derived through the meticulous calculation of the weighted average, utilizing a select group of representative individuals [9].

Discussions concerning the true essence of evaluation are seldom found, which often results in confusion with measurement and testing and lacks clear differentiation. A critical consequence of this absence is the lack of previous endeavors to establish universally applicable foundational evaluation principles and methodologies that cut across diverse disciplines, ultimately giving rise to significant ramifications.

Even within computer sciences and engineering, it is not uncommon for evaluators to generate greatly divergent evaluation outcomes for the same subject. These discrepancies can range from significant variations to the extent of yielding contradictory qualitative conclusions. An example of this phenomenon can be observed when using multiple widely recognized CPU benchmark suites to assess the performance of the same processor. This often leads to greatly divergent evaluation outcomes that are *incomparable* across different benchmarks. Such circumstances give rise to valid concerns surrounding the reliability, effectiveness, and efficiency of these approaches when appraising the subject that is critical to safety, missions, and businesses. Further details on this issue can be found in Section 5.

To the best of our knowledge, this article, for the first time, formally introduces the discipline of evaluatology, encompassing the science and engineering of evaluation. We present an all-encompassing concept, terminology, theory, and methodology framework for evaluation that can be universally applied across diverse disciplines if not all disciplines.

We highlight that the essence of evaluation lies in conducting deliberate experiments where a well-defined Evaluation Condition (EC) is applied to a well-defined subject. The purpose is to establish a well-defined Evaluation Model (EM). By measuring and/or testing this EM, we can then infer the impacts of the subjects being evaluated. Derived from the core essence of evaluation, we present five axioms as the foundational principles of evaluation theory. The five axioms focus on key aspects of evaluation outcomes, including true quantity (The first and second axioms), traceability of discrepancy (the third axiom), comparability (the fourth axiom), and estimate (the fifth axiom).

Based on the five evaluation axioms, we present the universal evaluation theories and methodologies from two distinct dimensions: evaluating a single subject and complex scenarios.

A well-defined EC serves as a prerequisite for meaningful comparisons and analyses of the subjects. We propose a universal hierarchical definition of an EC and identify five primary components of an EC from the top to the bottom.

In the process of evaluating subjects, it is of utmost importance to prioritize the use of the equivalent ECs (EECs) across diverse subjects. This means that in order to establish two EECs, it is crucial to ensure that the corresponding components within the same layer of the two ECs are equivalent. By maintaining equivalency at each layer, we can ensure fair and unbiased evaluations, enabling meaningful comparisons and assessments between different subjects.

In certain cases, achieving complete equivalence between two ECs at all levels can be a challenging or even unattainable task. In such cases, we propose a minimum requirement of ensuring uniformity in the most essential components of the two ECs, which we refer to as the least equivalent evaluation conditions (LEECs). To establish the LEECs, we identify the most governing component within an EC that must exhibit equivalency. This component, known as the evaluation standard, plays a crucial role in defining the LEECs.

We apply ECs with different levels of equivalency to diverse subjects to constitute EMs. An EM element refers to a specific point within the EM state space, and each EM element may have many independent variables. To eliminate confounding, we propose a new concept named a reference evaluation model (REM). An REM mandates that each element of an EM change only one independent variable at a time while keeping the other independent variables as controls. Subsequently, we utilize the measurement and/or testing to gauge the functioning of the

REM. Finally, from the amassed measurement and testing data of the evaluation systems, we then deduce the cause–effect impacts of the different subjects.

Addressing the complexities that arise in more intricate scenarios, we reveal that the key to effective and efficient evaluations in various complex scenarios lies in the establishment of a series of EMs that maintain transitivity.

In real-world settings, we refer to the entire population of real-world systems that are used to evaluate specific subjects as the *real-world evaluation system (ES)*. Assuming no safety concerns are present, the real-world ES serves as a prime candidate for creating an optimal evaluation environment, enabling the assessment of diverse subjects. However, there are several significant obstacles to consider, i.e., the presence of numerous confounding, the challenges of establishing an REM, prohibitive evaluation costs resulting from the huge state spaces, multiple irrelevant concurrent problems or tasks taking place, and the inclination to exhibit bias towards certain clusters within the EC state space.

We posit the existence of a *perfect EM* that replicates the real-world ES with utmost fidelity. A perfect EM eliminates irrelevant problems or tasks, has the capability to thoroughly explore and comprehend the entire spectrum of possibilities of an EC, and facilitates the establishment of REMs.

However, the perfect EM possesses huge state space, entails a vast number of independent variables, and hence results in prohibitive evaluation costs. To address this challenge, it is crucial to propose a pragmatic EM that simplifies the perfect EM in two ways: reducing the number of independent variables that have negligible effect and sampling from the extensive state space. A pragmatic EM provides a means to estimate the parameters of the real-world ES.

We put forth four fundamental issues in the evaluations and formally formulate the problems mathematically: ensure the transitivity of EMs; perform a cost-efficient evaluation with controlled discrepancies; ensure the evaluation traceability; connect and correlate evaluation standards across every discipline.

Building upon the science of evaluation, we formally define a benchmark as a simplified and sampled EC, specifically a pragmatic EC, that ensures different levels of equivalency. Based on this concept, we propose a benchmark-based universal engineering of evaluation across different disciplines, which we aptly term "benchmarkology". Fig. 1 presents the universal concepts, theories, and methodologies in Evaluatology.

The article is structured as follows: Section 2 elucidates the background. Section 3 introduces a comprehensive theoretical and methodological framework for evaluatology. Section 4 outlines the principles and methodologies of benchmarkology. Section 5 reviews the state-of-the-art and state-of-the-practice evaluations and benchmarks and expounds upon the imperative to cultivate the science and engineering of evaluation. Ultimately, Section 6 manifests the overarching conclusion.

2. Background

This section provides an overview of the background.

2.1. Basic concepts

This subsection presents several concepts, like individual, systems, populations, samples, variables, models, confounding, and control based on several undergraduate textbooks [2,10,11].

An individual can be defined as the object that is described by a given set of data. A *system* is an interacting or interdependent group of individuals, whether of the same or different kinds, forming a unified whole [12,13]. A system could be a recursive structure. That is to say, a high-level system could consist of an interacting or interdependent group of low-level systems, whether of the same or different kinds, forming a unified whole.

A *population* is the entire group of individuals or systems we wish to study and understand, while a *sample* represents a smaller subset of individuals or systems from the population [2]. A *variable* or *quantity* is any *property* of an individual or system. A *parameter* is a number that describes some property of the population, while a *statistic* is a number that describes some property of a sample. *Inference* is the process of drawing conclusions about a parameter of a population on the basis of the statistic of sample data [2].

According to [11], a function, denoted as f, is a rule that assigns a unique element, referred to as $f(x)$, from a set R to each element in a set D. In this context, the domain, denoted as D, refers to the set of all possible values for which the function is defined. On the other hand, the range of the function, denoted as $f(x)$, consists of all the possible values that $f(x)$ can take as x varies within the domain. The *independent variable* is represented by a symbol that encompasses any arbitrary number within the domain of the function. A *dependent variable*, represented by a symbol, is used to denote a number within the range of the function.

A *model* is a simplified version of a system that would be too complicated to analyze in full detail [10]. A model could be a physical model or a *mathematical model*. A mathematical model is a mathematical description, typically through *functions* or equations, of a system, the purpose of which is to understand the system and to make predictions about its behavior [11]. Throughout the remainder of this article, we will use the terms "system" and "model" interchangeably unless explicitly stated otherwise.

An *observational study* observes individuals or systems and measures variables of interest without any attempt to influence their responses, while an *experiment* is designed to deliberately impose conditions on individuals or systems to measure and analyze their responses [2].

In the realm of understanding cause and effect, it is crucial to rely on experiments rather than observational studies. Even if an observational study is based on a random sample, it still falls short in effectively measuring the impact of changes in one variable on another variable [2]. Experiments, on the other hand, provide us with compelling and conclusive data, making them the only source that truly convinces us of cause-and-effect relationships.

Confounding arises when two independent variables are associated in a manner that makes it challenging to differentiate their specific effects on a dependent variable. In other words, the influence of these independent variables becomes entangled, making it difficult to attribute specific impacts to each one. In such cases, the independent variable responsible for this confounding effect is referred to as a *confounding variable*. *Control* means keeping other independent variables that might affect the response the same [2], and the main purpose of a *control group* is to provide a baseline for comparing the effects of the other treatments.

2.2. Metrology

Metrology is the science of measurement and its applications [14]. In this section, we present a simplified yet systematic framework for understanding metrology concepts based on the works of [14,15]. To maintain conciseness, we focus only on the essential metrology concepts (see Fig. 2).

The essence of metrology lies in quantities and their corresponding measurements. A *quantity* is a property whose instances can be compared by ratio or only by order [14]. Furthermore, Psychologist Stanley Smith Stevens developed a well-known measurement classification with four levels based on empirical operations, mathematical group structure, and permissible statistics (invariant): nominal, ordinal (based on order [14]), interval, and ratio [16].[1]

[1] In the original article, Stanley Smith Stevens used the term "levels or scales of measurement". We have only used "levels" to avoid confusion with the specific meaning of "scales" in psychology

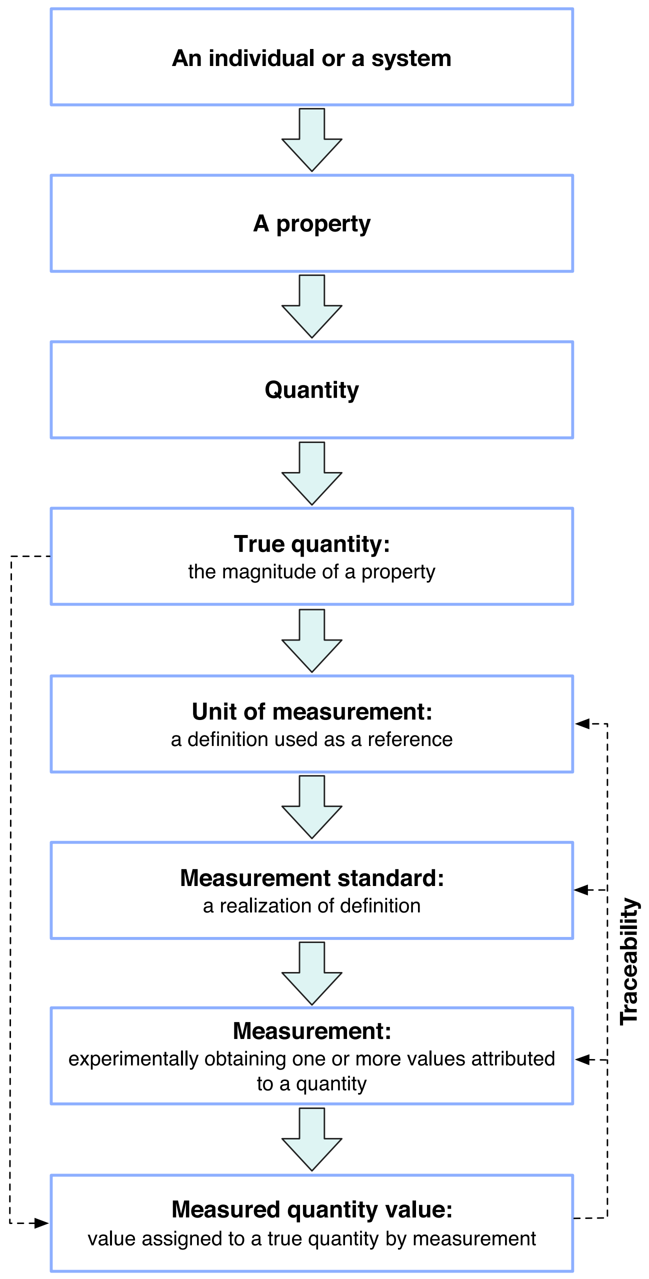

An individual or a system
A property
Quantity
True quantity:
the magnitude of a property
Unit of measurement:
a definition used as a reference
Measurement standard:
a realization of definition
Measurement:
experimentally obtaining one or more values attributed to a quantity
Measured quantity value:
value assigned to a true quantity by measurement
Traceability

Fig. 2. A simplified yet systematic conceptual framework for metrology [14,15].

A nominal level is the most basic form of measurement, where numbers are used as labels or type numbers to establish an equality relation. An ordinal level, on the other hand, involves ranking the items in a particular order. An interval level exhibits an equality of interval relation, where (1) the choice of a zero point is a matter of convention or convenience, (2) there is rank ordering, and (3) the scale remains invariant when a constant is added to all values, preserving the differences between them. A ratio level allows for all four types of relations: equality, rank-ordering, equality of intervals, and equality of ratios.

The international system of metrology encompasses seven fundamental quantities: time, length, mass, electric current, thermodynamic temperature, amount of substance, and luminous intensity [14].

Consistent with the definition of a quantity, the true value of a quantity represents the magnitude of a property or characteristic of an individual or system, e.g., a phenomenon, body, or substance that is independent of any observer. For example, it can be a specific circle's radius or a particular particle's kinetic energy within a given system [14,15]. For measurement, the true quantity value is an unknown measurement target [14].

In the field of measurement, the *unit of measurement* [15] plays a crucial role. It is a real scalar magnitude that is defined and adopted by convention. Its purpose is to allow for the comparison of quantities of the same kind.

Measurement standard [15] is a realization of the definition of quality. It is characterized by a stated metric value and an associated measurement uncertainty.

To establish a measurement standard, it is important to use a *measurement methodology* that is both repeatable (performed by the same team) and reproducible (performed by different teams). This ensures consistency and reliability in the reference for measurements. Such measurements can be conducted using *measuring instruments* or *measuring systems* [14], providing a reliable foundation for further analysis and comparison.

Measurement is experimentally obtaining one or more values attributed to a quantity and other relevant information [14]. Another widespread definition of measurement in the social sciences is "the assignment of numerals to objects or events according to some rule." [16], dating back to 1946. Quantity values obtained by the measurement are measured (quantity) values, representing the measurement results [14].

The hierarchy of measurement standards follows a progression from lower to upper levels, with increasing accuracy and cost. This progression starts from national measurement standards and extends to international standards. As a property of a measurement result, *measurement traceability* [14] establishes a connection between the result and a reference (measurement standards, measuring instruments, and measuring systems). This connection is established through a documented, unbroken chain of calibrations, with each calibration contributing to the measurement uncertainty. To ensure accuracy, each level of measurement standards in the hierarchy should be calibrated using a higher standard with greater precision.

2.3. Testing

A *test oracle* is a method used to verify whether an individual or system being tested has performed correctly during a specific execution [17]. Test oracles can include, but are not limited to, specifications, contracts, reference products, previous versions of the same product, and relevant performance or quality of service criteria [18].

As shown in Fig. 3, *testing* is the process of executing an individual or system to determine whether it (1) conforms to the specified behavior defined by the test oracles [19] (the first category) and/or (2) operates correctly within its intended environment as defined by the test oracles (the second category).

In the first category of testing, the test oracle compares the actual output with the specified output to identify incorrect behavior, which is considered a *failure* [19]. Another type of failure is often encountered in the second category of testing, where an individual or system fails to meet environmental constraints or falls outside the specified requirements. Examples of such failures include running out of memory, slow execution, and incompatibility with operating systems [17]. It is important to note that these two types of failures are not isolated incidents. Failures in the second category can lead to failures in the first category, such as running out of memory, which results in incorrect program execution.

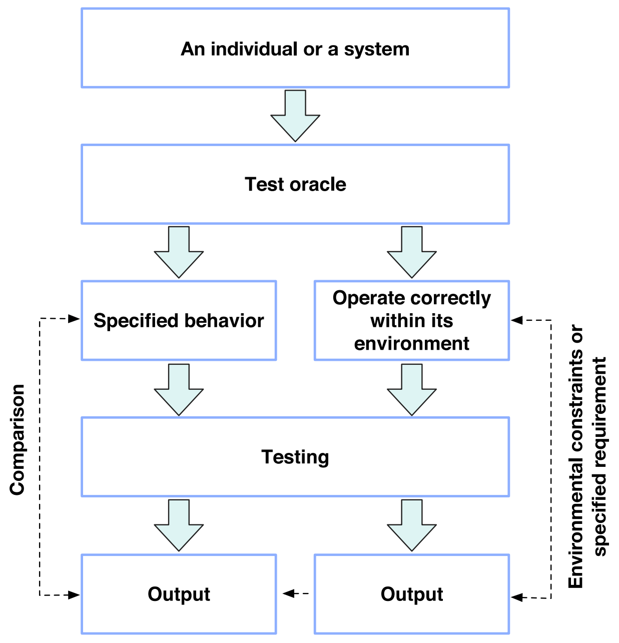

An individual or a system
Test oracle
Specified behavior
Operate correctly within its environment
Testing
Output
Output
Comparison
Environmental constraints or specified requirement

Fig. 3. A simplified yet systematic conceptual framework for testing [17,19].

3. The science of evaluation

Within this section and the forthcoming one, we first propose the universal evaluation concepts and terminologies. Then, we elucidate the theory and methodology governing the realm of evaluation, collectively referred to as evaluatology.

We present the science of evaluation from two distinct perspectives: evaluating a single subject and evaluating complex scenarios.

We postpone the exposition of the principles and methodology pertaining to benchmarkology, the benchmark-based engineering of evaluation, to Section 4.

3.1. Basic evaluation concepts and terminology

In this subsection, we will examine an illustrative case study in evaluation and meticulously analyze the fundamental *components* inherent in an evaluation. For the sake of convenience, henceforth, within this article, each case study shall be assigned a unique case ID for differentiation purposes.

In the first case, which we will call Case One, an organization is in the process of acquiring a computer. To make an informed decision, the organization decides to evaluate various computer options by executing its applications on each one. During this evaluation, the organization will collect extensive data on performance and energy efficiency.

Based on this data, the organization will then formulate an explicit or implicit function to express its preferences for different computers. This function will serve as a way to quantify and articulate the organization's priorities and requirements in terms of performance and energy efficiency. By doing so, the organization can make a well-informed decision and choose the computer that best aligns with its needs and preferences.

In any evaluation process, we refer to an individual or a system under scrutiny as a *subject*. In the context of Case One, the computers that are being considered for evaluation are the subjects.

Another important component of an evaluation is the presence of *stakeholders*. A stakeholder is defined as an entity that holds a stake of responsibility or interest in the subject matter. In this case, the stakeholders involved in the procurement of computers include the organization seeking to acquire the computers, the specific users, the designers responsible for creating the computer specifications, and the producers who manufacture the computers.

Each subject, in this case, the computer, has its own set of stakeholders who will render judgments based on the data collected from measurements. Metrology provides the foundational principles, methodology, and instruments for measurements within this process. It ensures that the measurements are accurate, reliable, and consistent, enabling the stakeholders to make informed decisions based on the gathered data.

In Case One, the prospect of measuring the mere attributes of a subject, such as its weight and power consumption, possesses a certain degree of utility. Nevertheless, such measurements fall considerably short of meeting stakeholders' evaluation requirements. The stakeholder seeks comprehension of the subject's effectiveness and efficiency when applying a specific condition or intervention to the subject. In this case, it is to execute the stakeholder's primary or forthcoming applications, which we informally label as an *evaluation condition (EC)*. The EC represents the third critical component of an evaluation, which shall be formally expounded upon subsequently.

Within this framework, an important question arises: How can organizations establish a framework to determine the preferences of distinct subjects when they exhibit varying levels of performance across different applications?

In the current state-of-the-practice, a more intuitive approach entails executing applications on computers sequentially. Subsequently, we proceed to measure the computers' performance when operating distinct applications individually. Following each execution, data is collected encompassing factors such as the duration of each application's execution and its corresponding energy consumption.

It is imperative to establish a function that can map the compiled measurement data to one or several composite evaluation metrics capable of capturing the stakeholders' concerns and interests. In the rest of this article, we refer to this function as *a value function*. Once the evaluation outcomes have been obtained, it becomes feasible to define a *reference subject* and its *reference evaluation outcome* against which the evaluation results of alternative subjects can be compared.

3.2. The essence of evaluation

From the aforementioned analysis in Section 3.1, it is evident that the challenge in evaluation arises from the inherent fact that evaluating a subject in isolation falls short of meeting the expectations of stakeholders. Instead, it is crucial to apply a well-defined EC that reflects the stakeholders' concerns or interests. By doing so, evaluation can be viewed as an intentional experiment that deliberately imposes a specific EC on the subject itself.

Based on the definitions provided in Section 2.1, when a subject is equipped with an EC, it forms an *evaluation system* (ES) or an *Evaluation Model* (EM). An evaluation model (EM) is a simplified version of an ES that would be too complicated to analyze in full detail [10].

Based on the analysis presented earlier, it becomes clear that *the core essence of evaluation lies in conducting deliberate experiments where Equivalent ECs (EECs) are applied to a diverse range of subjects, resulting in the establishment of equivalent EMs. Subsequently, we can effectively evaluate the subjects by measuring the equivalent EMs.*

Therefore, we formally define evaluation as *an experiment that applies EECs to diverse subjects and establishes equivalent EMs, enabling the measurement of these equivalent EMs, the inference of the subjects' impact, and the subsequent judgment of them.*

3.3. Five evaluation axioms

In this section, we present five evaluation axioms that are derived from the core essence of evaluation, serving as the foundational principles of evaluation theory. They focus on key aspects of evaluation outcomes, including true quantity (The first and second axioms), traceability of discrepancy (the third axiom), comparability (the fourth axiom), and estimate (the fifth axiom).

The First Axiom of Evaluation: The Axiom of the Essence of Composite Evaluation Metrics. This axiom declares that the essence of the composite evaluation metric either carries inherent physical significance or is solely dictated by the value function.

In nature, *a composite evaluation metric refers to a combined quantity that is constructed using base quantities and other quantities that possess physical significance.* If a composite evaluation metric does not carry inherent physical significance, the value function serves as a mechanism that maps base quantities and other quantities carrying physical meaning into a composite evaluation metric. The widespread acceptance of the composite evaluation metric relies on it being embraced by the community of evaluators.

The Second Axiom of Evaluation: The Axiom of True Evaluation Outcomes. This axiom declares that when a well-defined EC is applied to a well-defined subject, its evaluation outcomes, including its quantities and composite evaluation metrics, possess true values.

"The magnitude of a property of an individual phenomenon, body, or substance is associated with a true quantity" [14,15]. Additionally, each testing procedure yields a definitive outcome relative to its respective test oracle. Building upon this inference, it is reasonable to presume that when a well-defined subject is equipped with a well-defined EC, the quantities within the corresponding well-defined EM possess true values.

For a well-defined EM, each composite evaluation metric is derived from measurement and/or testing outcomes, utilizing a definite value function that translates the base quantities and other quantities into a composite evaluation metric. Consequently, the evaluation results are exclusively determined by the measurements and/or testing procedures employed. It is reasonable to assume that for a well-defined EM, its composite evaluation metric possesses a true value.

The first and second axioms are concerned with the true evaluation outcome. The first axiom provides the basis for defining value functions. With a well-defined value function, a well-defined EM possesses true quantities of evaluation outcomes.

The Third Axiom of Evaluation: The Axiom of Evaluation Traceability. This axiom declares that for the same subject, the divergence in the evaluation outcomes can be attributed to disparities in ECs, thereby establishing evaluation traceability. This axiom focuses on the traceability of discrepancies in the evaluation outcomes.

For the same subject, this axiom is deemed rational as disparities in evaluation outcomes can be rationalized as the consequence of variations in the ECs. In the absence of this axiom, the differences observed in evaluation outcomes would be inexplicable, contradicting our scientific and engineering intuitions.

The Fourth Axiom of Evaluation: The Axiom of Comparable Evaluation Outcomes. This axiom declares when each well-defined subject is equipped with EECs, their evaluation outcomes are comparable. It goes without saying this axiom is related to the comparability of the evaluation outcomes.

Only when each EC is well-defined, and two ECs achieve complete equivalence at all levels can we refer to them as EECs. When each well-defined subject is equipped with EECs, its evaluation outcomes possess true values. Additionally, when well-defined subjects are subjected to EECs, their evaluation outcomes accurately reflect the impacts of different subjects under the same conditions, making them comparable.

The Fifth Axiom of Evaluation: The Axiom of Consistent Evaluation Outcomes. This axiom asserts that when a well-defined subject is evaluated using different samples from a population of ECs, their evaluation outcomes consistently converge towards the true evaluation outcomes of the population of ECs. This axiom provides an estimate of the true evaluation outcomes under the population of ECs.

According to the Second Axiom of Evaluation, when a well-defined subject is equipped with a well-defined population of ECs, the resultant EM possesses the true evaluation outcomes. When a sample is taken from a population of ECs, it serves as an approximation of the entire population. As a result, different samples yield consistent evaluation outcomes that gradually converge towards the evaluation outcomes of the entire population of ECs. This convergence is influenced by the sample's ability, which is determined by the chosen sampling policy, to represent the underlying characteristics of the population accurately.

3.4. Basic evaluation methodology

This section outlines the fundamental methodology for evaluating a single subject. Drawing upon the discussion of the essence of evaluation in Section 3.2, we propose a rigorous evaluation methodology to determine the impacts of the subjects as follows.

We create equivalent ECs (EECs) and apply EECs to diverse subjects to constitute equivalent EMs. An EM element refers to a specific point within the EM state space, and each EM element may have many independent variables. To eliminate confounding, we propose a new concept named a reference evaluation model (REM). An REM mandates that each element of an EM change only one independent variable at a time while keeping the other independent variables as controls. Subsequently, we utilize the measurement to gauge the functioning of the REM. Finally, from the amassed measurement data of the evaluation systems, we then deduce the cause–effect impacts of the different subjects.

In this methodology, we emphasize five essential steps to ensure a comprehensive evaluation, as shown in Fig. 4. These steps are crucial in accurately determining the impacts of the subjects.

The first step is to establish a rigorous definition of an EC. According to The Second and Three Axioms of Evaluation, when a well-defined subject is equipped with a well-defined EC, its evaluation outcomes possess true values; when the same subject is equipped with different ECs, any divergence in the evaluation outcomes can be attributed to disparities in these ECs. Therefore, the key focus in this phase is to clearly present a well-defined EC.

The second step involves the establishment of EECs. As outlined in the Fourth Axiom of Evaluation, when EECs are applied to diverse subjects, their evaluation outcomes become comparable. Since the primary objective of evaluation is to compare different subjects, the establishment of EECs becomes an essential step in the process.

The third step involves the elimination of confounding variables. Given that each element of an EM consists of multiple independent variables, it becomes essential to establish an REM. An REM serves as a controlled evaluation environment where only one independent variable is altered at a time while the other independent variables remain constant as the control. This approach helps in isolating the effects of individual variables and ensures a more accurate evaluation of the subject's performance.

The fourth step is to define the value functions that map the base quantities and other quantities to composite evaluation metrics that represent the stakeholders' primary concerns or interests. According to the first axiom of evaluation, the essence of the composite evaluation metric either carries inherent physical significance or is solely dictated by the value function. So, when defining a value function, it is crucial to make it become the consensus of the community.

Finally, we utilize the measurement to gauge the functioning of the REM. From the amassed measurement data of the REM, we then deduce the cause–effect impacts of the different subjects.

3.5. Basic evaluation theory

This subsection presents the basic evaluation theory, including the hierarchical definition of an EC, universal concepts across different disciplines, the establishment of EECs, LLECs, evaluation standards, and the establishment of an REM.

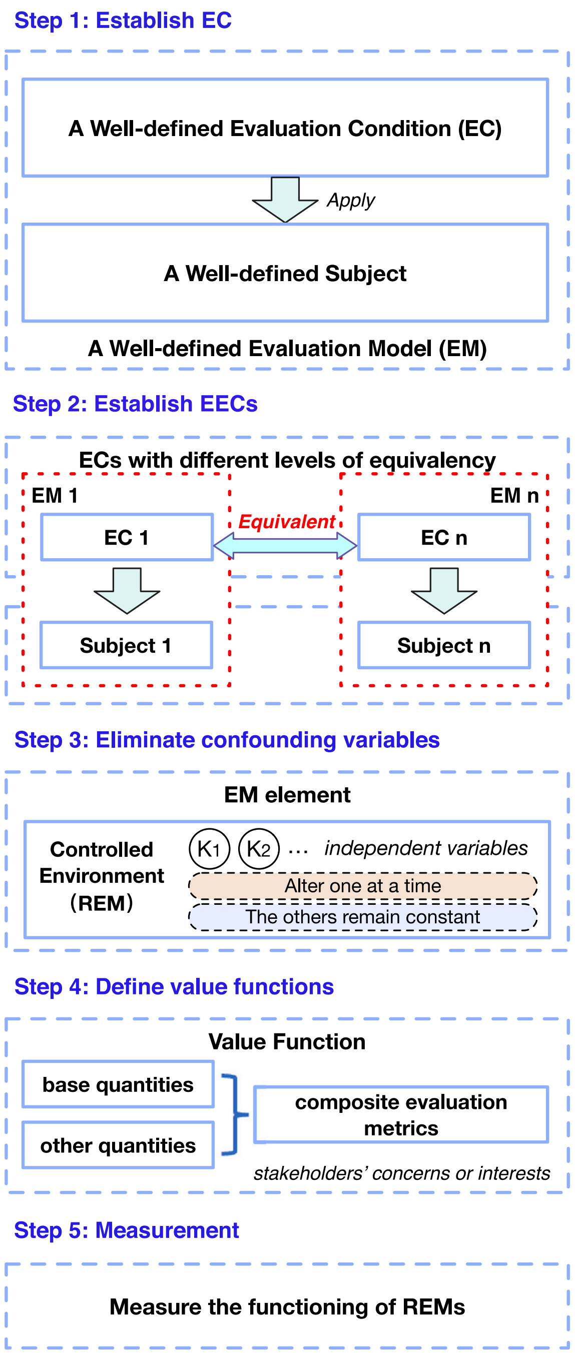

Step 1: Establish EC
A Well-defined Evaluation Condition (EC)
Apply
A Well-defined Subject
A Well-defined Evaluation Model (EM)
Step 2: Establish EECs
ECs with different levels of equivalency
EM 1
EM n
EC 1
Equivalent
EC n
Subject 1
Subject n
Step 3: Eliminate confounding variables
EM element
Controlled Environment (REM)
K1
K2
… independent variables
Alter one at a time
The others remain constant
Step 4: Define value functions
Value Function
base quantities
other quantities
composite evaluation metrics
stakeholders' concerns or interests
Step 5: Measurement
Measure the functioning of REMs

Fig. 4. Basic evaluation methodology.

3.5.1. *The hierarchical definition of an EC*

In the preceding subsection, we deduced that there exists solely one feasible approach to evaluation: applying the EECs to diverse subjects and establishing an REM. Regrettably, even within a rudimentary evaluation setting such as Case One, an EC has multifarious components.

To address the above challenges, in this subsection, we propose a hierarchy definition of an EC. We start defining an EC from the problems or task spaces that these stakeholders face and need to address with the following two reasons. First, the concerns and interests of the relevant stakeholders are at the core of the evaluation. These concerns and interests are best reflected through the problems or tasks they must face and resolve, which provide a reliable means to define an EC. Second, utilizing the same problem or task can ensure the comparability of evaluation outcomes.

Taking Case One as an example, we observe that in Case One, an EC encompasses numerous constituents. Notably, we identify four primary components of an EC from the top to the bottom. The first top component is a set of equivalent definitions of problems or tasks. While the problem or task itself serves as the foundation for the evaluation process, it cannot solely serve as the evaluation itself because the problem or task is often abstract and requires further instantiation to determine its specific parameters. The second component is the set of a collective of equivalent problem or task instances, each of which is instantiated from the element of the first component. Different from the first component, an equivalent problem or task instance is specific and could serve as the evaluation directly.

After a problem or task instance is proposed, it is necessary to figure out a solution. The third component consists of the algorithms or algorithm-like mechanisms, each of which provides the solution to a specific problem or task instance. An algorithm-like mechanism refers to a process that operates in a manner similar to an algorithm. This term is proposed because, in numerous disciplines, such as social and biological sciences, it is not currently feasible to formulate mathematical algorithms explicitly. These domains often involve complex and nuanced phenomena that defy precise mathematical modeling.

The fourth component encompasses the implementation of an algorithm or instantiation of an algorithm-like mechanism. Its implementation or instantiation involves understanding the algorithm or the algorithm-like mechanism and implementing it in a specific system. This process ensures that the algorithm or the algorithm-like mechanism can effectively and efficiently solve the intended problem instance or perform the desired task instance within the given context.

In addition to the four components of an EC that we discussed above, other components can be involved in the other complex evaluation scenarios, which we will discuss later.

3.5.2. *Universal concepts across different disciplines*

In Section 3.1, we conducted an examination of the fundamental constituents of an evaluation, which encompass “subjects” and ”evaluation conditions”. Additionally, we put forth definitions for various fundamental concepts, namely “subject”, ”stakeholders”, and ”value functions”. Through this analysis, we unveiled that the core nature of evaluation is to intentionally apply EECs to diverse subjects and establish an REM to infer the impact of the subjects for judgments.

However, given the multiplicity of evaluation scenarios, two critical questions must be addressed: (1) Do these concepts suffice for diverse scenarios? (2) Can we formulate a comprehensive and universally applicable conceptual framework? In this subsection, we delve deeper into various evaluation cases across diverse disciplines, with the primary aim of enhancing our comprehension of the evaluation process.

Evaluating an AI algorithm

In Case Two, the objective is to evaluate an AI algorithm, specifically focusing on an Image Classification task as a case study. Real-world images are gathered and annotated with accurate labels, such as a cat or dog. A portion of these images is randomly selected to construct the training, validation, and test datasets based on a designated percentage. To assess the image classification algorithm (the subject in this case), it must be implemented on a computer system utilizing a specific programming framework, such as PyTorch or TensorFlow.

During the evaluation process, the test data is provided to the algorithm, which generates an output. This output is then compared to the ground truth labels. In Case Two, the ground truth can be considered as a test oracle. Additionally, measurements are collected for each run of the evaluation process.

The fundamental evaluation process in Case Two bears a resemblance to that of the baseline case. However, there exist three notable distinctions. Firstly, an EC in Case Two differs. The presence of a dataset labeled with ground truth constitutes a vital aspect of an EC.

A dataset labeled with the ground truth represents a specific instance of a problem or task. Regrettably, in Case Two, it is impractical to define a problem or task mathematically. Hence, diverse problem or task instances are devised in an ad-hoc manner, for example, selecting images randomly to form training, validation, and test datasets based on a predetermined percentage. This approach may introduce biases.

Secondly, both the subject and the algorithms must be implemented on a computer, prompting the introduction of another concept called "a support system" to elucidate this facet, which may assume diverse manifestations in alternative evaluation scenarios. The support system represents an additional essential constituent of an EC. Therefore, in Case Two, we introduce two novel concepts: "the support system" and "the subject instantiation". Fig. 5 shows a hierarchy definition of an EC.

Thirdly, in Case Two, upon feeding the algorithm with the test data, it generates an output that is then compared with the ground truth, also known as the test oracle. Apart from measurements, there exist other forms of activities in the evaluation process, namely testing, as expounded upon in Section 2.3. Hence, in this instance, we modify the essence and definition of evaluation, as delineated in Section 3.2, as follows. The essence of an evaluation lies in "conducting deliberate experiments where EECs are applied to a diverse range of subjects, resulting in the establishment of equivalent EMs. Subsequently, we can effectively evaluate the subjects by measuring and/or testing the equivalent EMs". We formally define evaluation as "an experiment that applies EECs to diverse subjects and establishes equivalent EMs, enabling the measurement and/or testing of these equivalent EMs, the inference of the subjects' impact, and the subsequent judgment of them".

Drug and policy evaluations

The Third Case (Case Three) and the Fourth Case (Case Four) exhibit similarities, as they involve evaluating a drug and evaluating a policy aimed at addressing drug addiction within a community. In these cases, the subject refers to a specific drug or a policy aimed at addressing drug addiction intervention, while the support system encompasses the participants targeted by these interventions.

When comparing Cases Three and Four with Cases One and Two, we have made three observations. Firstly, the specific problem or task instances could be defined in a literal manner. Case Three focuses on the cure of a specific illness, whereas Case Four aims to address drug addiction and improve the overall well-being of individuals within a designated community. Regrettably, we currently lack a mathematical understanding of these problems or tasks, making it challenging to provide a mathematical definition of the abstract problem or task. Nevertheless, even if the problems can only be defined in a literal sense, having detailed and comprehensive definitions of problem or task instances, as well as a profound comprehension of the interrelationships between different problem or task instances (from biological or social perspectives), proves advantageous for the purposes of knowledge reuse and sharing. Furthermore, it is anticipated that in the future, we will strive to gain a deeper understanding of the connections between various diseases or social issues, potentially through mathematical means.

The second observation revolves around the distinctions of the support systems found in Cases Two, Three, and Four, specifically referring to the substantial variability in conditions within the target participants. Evaluations commonly utilize a methodology known as randomized controlled trials (RCT), which serves to eliminate confounding.

In practical application, Randomized Controlled Trials (RCTs) are widely recognized as the gold standard for conducting evaluations. In an RCT, subjects and support systems in the treatment group and control group are randomly assigned. This random assignment helps to minimize confounding variables that may arise from differences in support systems. Additionally, the allocation of participants to either the treatment or control group is kept concealed from the evaluator and relevant stakeholders.

Lastly, when it comes to Cases Three and Four, the third component of an EC often lacks a mathematical-form algorithm, although it relies on a scientifically valid mechanism that includes biological, social, or psychological interactions. That is the right reason for us proposing the term "algorithm-like mechanism", which we have explained before. Moreover, Case Four presents a significantly more intricate situation compared to Case Three, as the instantiation of the algorithm-like mechanism encounters challenges in maintaining consistent quality due to diverse factors, such as different attitudes towards interventions and varying levels of communication skills that hold considerable importance.

In summarizing, across various evaluation scenarios in different disciplines, such as Cases One, Two, Three, and Four, it is possible to develop a comprehensive conceptual framework that can be universally applied.

3.5.3. The establishment of EECs

To lay the groundwork for the formalization of an EC, it is essential to establish a clear understanding of some crucial notations. The notations E', S, and U represent three crucial components. Specifically, E' represents the problem or task space. S represents the support system space. Finally, U represents the subject space. e', s, u is an element of E', S, and U respectively. We note $e' \in E'$, $s \in S$, $u \in U$.

In addition to the aforementioned notations, we also define several other fundamental notations. For each problem or task, $e'_i \in E'$, there is a set of problem or task instances noted as E_i. For all problems or tasks in E', there is a collection of a set of problem or task instances, which we noted as $SE = (e'_i, E_i)$. We use the division notation SE/E' to denote E. E can be defined as the union of all E_i.

We introduce the notation SA' to represent the algorithm-like mechanism space. This space, denoted as SA', consists of a set of algorithm-like mechanisms that are associated with each problem or task instance. Specifically, for a given problem or task e'_i in the problem or task space E', and for each instance e_{ij} in the corresponding instance space E_i, we define a set of algorithm-like mechanisms as $SA' = (e'_i, e_{ij}, A'_{ij})$.

To represent the algorithm-like mechanism space SA' in relation to the problem or task space E' and the problem or task instance space E, we use the division notation $SA'/E'/E$ to denote A'. A' is a union of all A'_{ij}.

We introduce the notation SA to represent the instantiations of the algorithm-like mechanism space. This space, denoted as SA, consists of a set of instantiations of algorithm-like mechanisms that are associated with each problem or task instance, algorithm-like mechanism, and support system. Specifically, for a given problem or task e'_i in the problem or task space E', for each instance e_{ij} in the corresponding instance space E_i, for each algorithm-like mechanism a'_{ijk} in the algorithm-like mechanism space A'_{ij}, and for each support system s_l in the support system space S, we define a set of instantiations of algorithm-like mechanisms as $SA = (e'_i, e_{ij}, a'_{ijk}, s_l, A_{ijkl})$.

To represent the instantiations of the algorithm-like mechanism space SA in relation to the problem or task space E', the problem or task instance space E, the algorithm-like mechanism space A', and the support system space S, we use the division notation $SA/E'/E/A'/S$ to denote A. A is a union of all A_{ijkl}.

By introducing these notations, we establish a comprehensive framework that allows us to delineate the various components of an EC and their respective roles. This formalization enhances our understanding of the key components and their relationships within the EC framework.

We can formalize an EC as $C = E' \times E \times A' \times A \times S$.

In the realm of EC spaces, the concept of EECs plays a significant role. Two EC spaces, denoted as C_1 and C_2, are considered to be EECs if and only if there exists a bijection, denoted as β, between the two spaces: $\beta : C_1 \mapsto C_2; \beta^{-1} : C_2 \mapsto C_1$. This equivalence is denoted as $C_1 \sim C_2$.

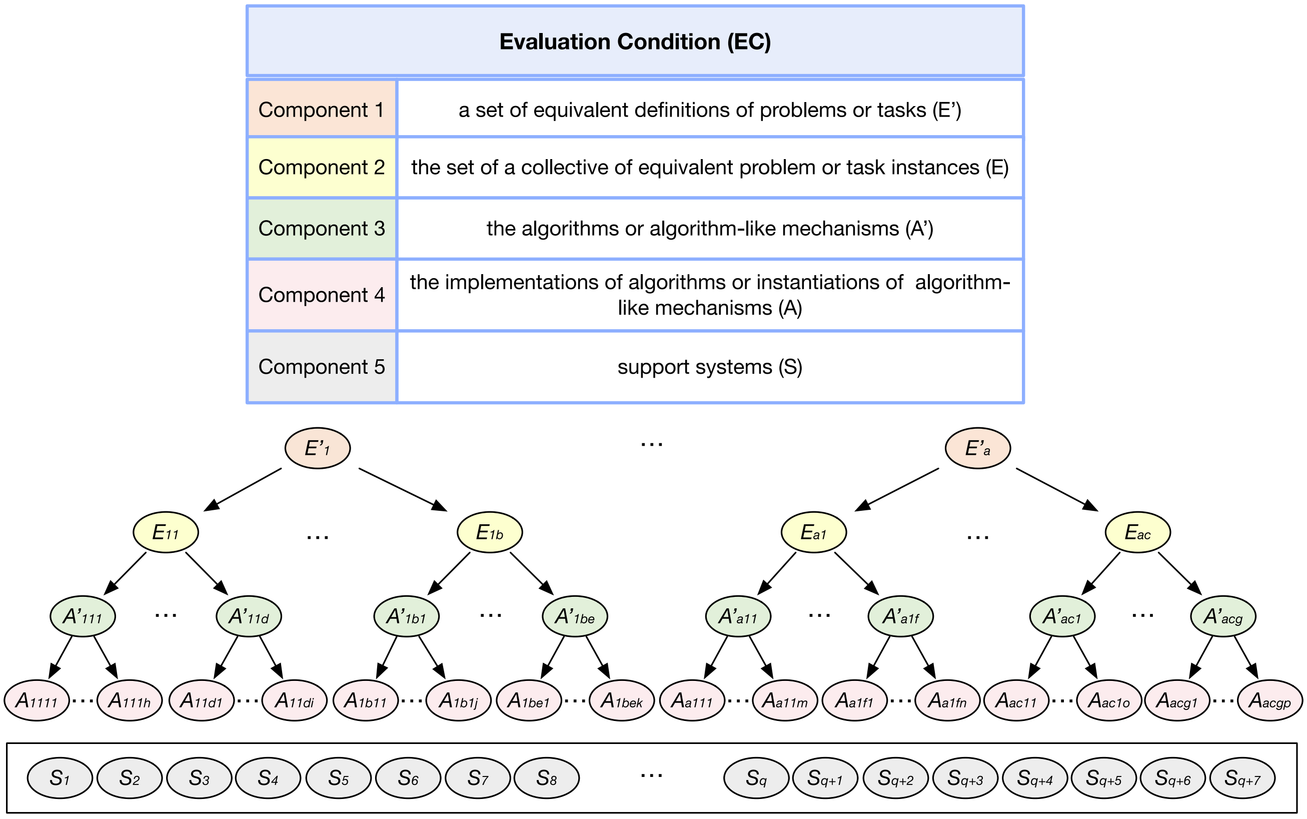

Evaluation Condition (EC)
Component 1 | a set of equivalent definitions of problems or tasks (E')
Component 2 | the set of a collective of equivalent problem or task instances (E)
Component 3 | the algorithms or algorithm-like mechanisms (A')
Component 4 | the implementations of algorithms or instantiations of algorithm-like mechanisms (A)
Component 5 | support systems (S)
E'1
E'a
E11
E1b
Ea1
Eac
A'111
A'11d
A'1b1
A'1be
A'a11
A'a1f
A'ac1
A'acg
A1111
A111h
A11d1
A11di
A1b11
A1b1j
A1be1
A1bek
Aa111
Aa11m
Aa1f1
Aa1fn
Aac11
Aac1o
Aacg1
Aacgp
S1 S2 S3 S4 S5 S6 S7 S8 … Sq Sq+1 Sq+2 Sq+3 Sq+4 Sq+5 Sq+6 Sq+7

Fig. 5. The hierarchical definition of an EC.

3.5.4. LEECs and evaluation standard

In certain cases, establishing EECs can be a challenging task. It proves to be an arduous or unattainable task to ensure the equivalence of two ECs at all levels. However, it is crucial to ensure the relative comparability of evaluation outcomes.

In this section, we propose the concept of the least equivalent evaluation conditions (LEECs) as the foundation to attack this challenge. In the event where we are unable to guarantee or establish the equivalence of two ECs at all levels in the hierarchy, we propose a minimum requirement of ensuring uniformity in the most essential components of the two ECs, which we refer to as the least equivalent evaluation conditions (LEECs).

We propose the establishment of LEECs at the levels of the first and second top components of ECs. Firstly, the first and second top components of an EC serve as the foundation upon which the other two lower-level components are derived. Therefore, they provide the most primitive components when setting ECs. Secondly, to enable effective comparison of evaluation outcomes, it is crucial to establish equivalence between the first high-level components of two ECs. If the first high-level components are not equivalent, the evaluation outcomes cannot be compared reliably. Thirdly, relying solely on the first component may not provide enough specificity and certainty. To address this, in addition to the equivalence of the first high-level component, it is necessary for two ECs to possess two sets of definite and solvable problem or task instances that are equivalent.

In certain situations, it is possible to relax the requirement of strict equivalency between the second high-level component if the scale of the problem or task instances can be defined. In such cases, we can consider a scenario where two ECs have the same problem or task but differ in the scales of problem or task instance as LEECs.

We formally define LEECs as follows:

For two ECs, denoted as $C_1 = E_1' \times E_1 \times A_1' \times A_1 \times S_1$ and $C_2 = E_2' \times E_2 \times A_2' \times A_2 \times S_2$, if there is equivalence between their first two subspaces (E' and E), that is, if and only if there is a bijection, denoted as $\hat{\beta}$, between $E_1' \times E_1$ and $E_2' \times E_2$, we can establish that they are LEECs, denoted as $C_1 \approx C_2$. In other words, $\hat{\beta}$ is a function mapping from $E_1' \times E_1$ to $E_2' \times E_2$, and its inverse function $\hat{\beta}^{-1}$, maps from $E_2' \times E_2$ to $E_1' \times E_1$, denoted as $\hat{\beta} : E_1' \times E_1 \mapsto E_2' \times E_2; \hat{\beta}^{-1} : E_2' \times E_2 \mapsto E_1' \times E_1$.

To effectively define the least equivalent ECs (LEECs), it is crucial to identify the most governing component of ECs that must exhibit equivalency. This component, known as the *evaluation standard*, plays a crucial role in defining the LEECs. By establishing and adhering to this evaluation standard, we can ensure that the evaluation outcomes are relatively comparable.

An evaluation standard should embody the characteristics that are solvable, definite, and equivalent (abbreviated as SDE). First, it should be amenable to a solvable framework, employing specific mechanisms. These mechanisms could encompass mathematical steps executed in a mechanical fashion [20] or incorporate biological, social, and psychological mechanisms and interactions. It is noteworthy that algorithms can be regarded as specific applications of mathematical steps executed in a mechanical manner. If an evaluation standard does not lend itself to a solvable framework, it becomes essentially meaningless. Second, it should possess definiteness, whereby there exists a unanimous understanding among evaluators without any uncertainty. Third, it should exhibit equivalence across multiple evaluators, ensuring consistency among their assessments.

We propose the establishment of an evaluation standard at the level of the definition of an individual problem or task instance. There are several valid reasons for this approach.

Firstly, the first and second high-level components define LEECs, and they are the cornerstone of the evaluation conditions. These two high-level components serve as the foundation upon which the other two low-level components are derived.

Secondly, the first high-level component, which pertains to the definition of a problem or task, is not definite and specific, as it may encompass a population of different instances. To be precise, an equivalent, definite, and solvable problem or task instance qualifies as “an

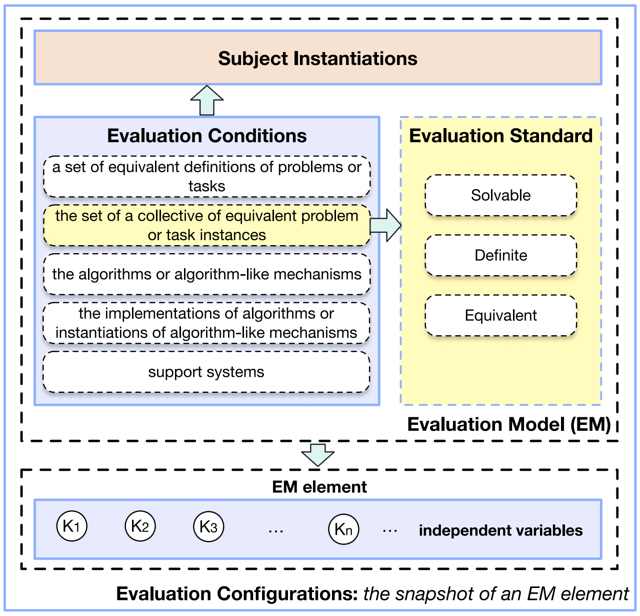

Subject Instantiations
Evaluation Conditions
a set of equivalent definitions of problems or tasks
the set of a collective of equivalent problem or task instances
the algorithms or algorithm-like mechanisms
the implementations of algorithms or instantiations of algorithm-like mechanisms
support systems
Evaluation Standard
Solvable
Definite
Equivalent
Evaluation Model (EM)
EM element
K1
K2
K3
···
Kn
···
independent variables
Evaluation Configurations: the snapshot of an EM element

Fig. 6. The relationships among evaluation configurations, evaluation conditions (C), Evaluation models (M), and evaluation standards. Please note that $C = M/U$.

evaluation standard". This definition serves as the basis for conducting evaluations on the subjects. Fig. 6 shows the relationships among the evaluation standard, evaluation conditions, evaluation configuration, and evaluation standard.

We discuss the subtle differences between LEECs and evaluation standards. LEECs are defined on the first and the second high-level components, while the evaluation standard is only defined at the second high-level component, and they are closely related but with distinct implications. Their shared objective is to guarantee the comparability of evaluation outcomes. The aim of LEECs is to ensure the least equivalence between two ECs, while the evaluation standard is to provide the most governing component of an EC that ensures the comparability of the evaluation outcomes. Second, LEECs imply a state space, while an evaluation standard is an evaluation criterion.

3.5.5. The establishment of an REM

Based on the notations defined in Section 3.5.3, we formalize an EM as $M = C \times U = E' \times E \times A' \times A \times S \times U$.

An element of an EM, denoted as $m \in M$, can be expressed as $m = (c, u) = (e', e, a', a, s, u)$. Here, $e' \in E'$ represents a given problem or task, $e \in E$ represents a specific instance of the problem or task, $a' \in A'$ represents a particular algorithm-like mechanism, $a \in A$ represents an instantiation of the algorithm-like mechanism, $s \in S$ represents a support system, and $u \in U$ represents a subject instantiation.

In addition, we will define how to establish an REM. We assume that an EM element, denoted as $m \in M$, is made of n' independent variables, rewritten as $m = (k_1, \dots, k_{n'}) = (e', e, a', a, s, u)$. We note $M = K_1 \times \cdots \times K_n = C \times U = E' \times E \times A' \times A \times S \times U$. Please bear in mind that the number of variables n is greater than n'.

For each EM element, represented as $m = (k_1, \dots, k_{n'})$, we follow a specific methodology in which only one independent variable at a time, from the set $k_1, k_2, \dots, k_{n'}$, is allowed to vary while keeping the remaining variables constant. This controlled experimentation approach is referred to as an REM, as defined in Section 3.4.

We define the evaluation cost of an EM or ES as the costs of constructing, traversing, and assessing its corresponding REM, where $\|M\|$ stands for the capacity of an EM or ES and μ stands for a constant coefficient:

$$\text{cost}(M) = \mu\|M\| = \mu\|K_1\| \times \|K_2\| \times \cdots \times \|K_n\|$$

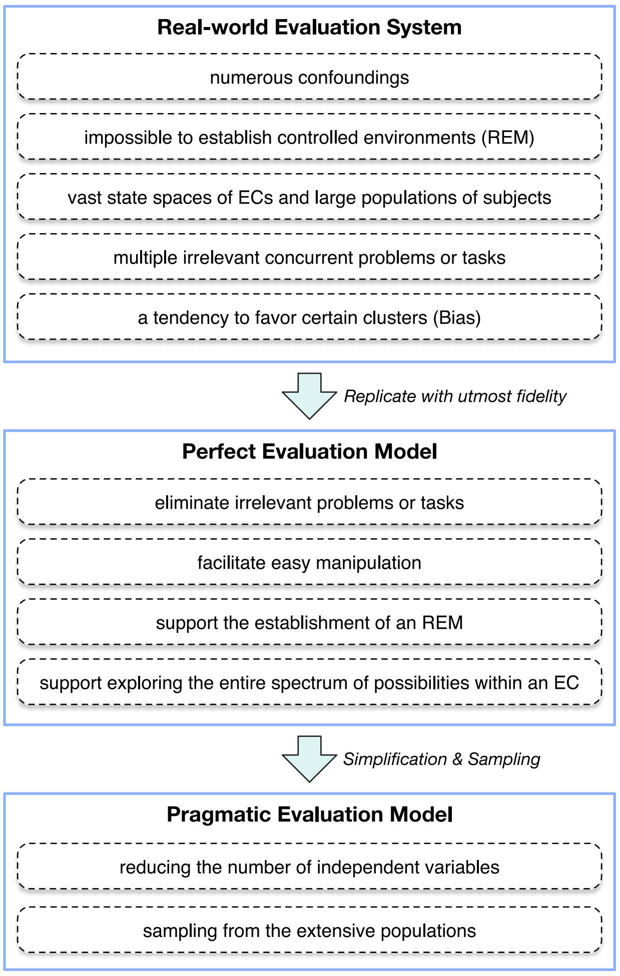

Real-world Evaluation System
numerous confoundings
impossible to establish controlled environments (REM)
vast state spaces of ECs and large populations of subjects
multiple irrelevant concurrent problems or tasks
a tendency to favor certain clusters (Bias)
Replicate with utmost fidelity
Perfect Evaluation Model
eliminate irrelevant problems or tasks
facilitate easy manipulation
support the establishment of an REM
support exploring the entire spectrum of possibilities within an EC
Simplification & Sampling
Pragmatic Evaluation Model
reducing the number of independent variables
sampling from the extensive populations

Fig. 7. Universal evaluation methodology in complex scenarios.

3.6. Universal evaluation methodology in complex scenarios

From the revealed essence of the evaluation, it seems that performing an evaluation is straightforward. Unfortunately, in reality, there are evaluation scenarios with different levels of complexity. Fig. 7 presents a universal evaluation methodology in complex scenarios.

We refer to the entire population of real-world systems that are used to evaluate specific subjects as the *real-world ES*. Assuming no safety concerns are present, the real-world ES serves as a prime candidate for the assessment of the subjects of interest. Unfortunately, there are five significant obstacles to consider when assessing diverse subjects within a real-world ES.

Firstly, the presence of numerous confounding in the real-world ES poses a considerable challenge. It is often difficult, if not impossible, to completely eliminate these confounding. They can complicate the evaluation process by introducing variables that make it challenging to isolate the effects of different independent variables.

Secondly, manipulating the real-world ES is a formidable task, making it virtually impossible to establish controlled environments, known as REM, for evaluating subjects. Additionally, the interconnected nature of subjects, support systems, and other components of ECs further complicates the establishment of an REM.

Thirdly, the vast state spaces of ECs and the large populations of subjects result in high evaluation costs. The sheer scale of these systems makes it expensive and time-consuming to thoroughly performing assessment and analysis.

Fourthly, within the real-world ES, multiple irrelevant concurrent problems or tasks may be taking place simultaneously, which may not directly align with the subject being assessed. This further adds complexity to the evaluation process and introduces confounding.

Lastly, it is important to acknowledge that the real-world ES, regardless of the nature of its problems or tasks, tends to exhibit a bias towards certain clusters. This bias can manifest through problem or task instances, algorithm-like mechanisms, and their instantiations. However, this bias towards specific groups can limit our ability to fully explore and understand the entire range of possibilities available to us.

When assessing a specific quantity of interest in an evaluation experiment, it is crucial to closely examine the relationship between the quantity of the EM and the corresponding quantity of the real-world ES. The ratio between these two quantities serves as a key indicator of accuracy. Ideally, a ratio closer to 100% signifies higher accuracy in the EM's modelling of the real-world ES.

In our research, we propose the concept of a "perfect EM" that aims to replicate the real-world ES with the highest level of fidelity, achieving a remarkable ratio of 100%. In theory, a perfect EM would possess several characteristics that enhance the evaluation of subjects within the EC framework.

Firstly, it would eliminate irrelevant problems or tasks that may be used to establish ECs, ensuring that the evaluation focuses on specific and directly applicable contexts. This targeted approach would enhance the relevance and applicability of the evaluation process.

Secondly, a perfect EM would facilitate easy manipulation, allowing for the free and artificial configuration of different evaluation settings. This flexibility would enable researchers to explore various scenarios and assess subjects under a range of conditions, enhancing the depth and breadth of the evaluation process.

Thirdly, a perfect EM would support the establishment of an REM, effectively eliminating confounding. By isolating and controlling variables of interest, researchers could gain more accurate insights into the impact of specific factors on the subjects being evaluated.

Furthermore, a perfect EM would have the capability to thoroughly explore and understand the entire spectrum of possibilities within an EC. This would include problem or task instances, algorithms-like mechanisms, and their instantiations. By encompassing this comprehensive range, researchers could gain deeper insights into the behavior and performance of subjects within the EC framework.

However, it is important to note that achieving a truly "perfect EM" may be challenging, if not impossible. The real-world ES is complex and dynamic, and replicating it with absolute fidelity is a monumental task. While we can strive to create more accurate and representative evaluation environments, it is crucial to recognize the inherent limitations and constraints that exist in the real world.

Nevertheless, by considering the concept of a perfect EM and its accompanying characteristics, we can strive to improve the evaluation of subjects within the EC framework and enhance our understanding of their performance within real-world contexts.

The characteristics of the perfect EM, such as encompassing large populations of problem or task instances, algorithm-like mechanisms, instantiations, and support systems, as well as a vast number of independent variables, can lead to significant evaluation costs. However, to address this challenge, it is important to propose a pragmatic EM that simplifies the perfect EM in two key ways.

Firstly, to reduce the evaluation costs associated with a large number of independent variables, it is crucial to identify and focus on the variables that have a significant impact on the evaluation outcomes. By identifying and prioritizing these variables, researchers can streamline the evaluation process and allocate resources more efficiently. Negligible variables that have a minimal effect can be excluded or controlled for, reducing complexity and costs. It is worth emphasizing that the simplification involved in creating a pragmatic EM will inevitably lead to a decrease in the accuracy of the evaluation model.

Secondly, sampling techniques can be employed to manage the extensive populations of problem or task instances, such as algorithm-like mechanisms, their instantiations, and support systems. Rather than evaluating every single possibility, researchers can select representative samples that capture the diversity and range of the population. This approach allows for a more manageable evaluation process while still maintaining a good level of coverage and representation.

In nature, a pragmatic EM is a subject equipped with a simplified and sampled EC. It can be considered as a sample of a perfect EM without taking into account any potential decrease in its accuracy. In order to measure the extent to which the statistics of a pragmatic EM can infer the parameters of a perfect EM, we employ the criterion of confidence interval and confidence level. The confidence level provides us with the probability that the estimated parameters of a perfect EM fall within a specific range of values. Meanwhile, the confidence intervals establish a range of values within which we can reasonably expect the true parameters of a perfect EM to fall. By utilizing these statistical measures, we can assess the degree of alignment between the statistics of a pragmatic EM and the parameters of a perfect EM. This allows us to gauge the effectiveness and validity of the pragmatic EMs.

By implementing these simplifications in a pragmatic EM, researchers can strike a balance between comprehensiveness and feasibility. The pragmatic EM allows for a more practical and efficient evaluation of subjects within the EC framework, mitigating the challenges posed by evaluation costs and the complexity of the perfect EM.

In representing different ECs, we use specific symbols. The symbol C_r denotes the EC in a real-world ES (a real-world EC), which can be calculated as $E_r' \times E_r \times A_r' \times A_r \times S_r$. Similarly, the EC in a perfect EM (a perfect EC) is denoted by C_p, calculated as $E_p' \times E_p \times A_p' \times A_p \times S_p$. Lastly, the EC in a pragmatic EM (a pragmatic EC) is represented by C_g, calculated as $E_g' \times E_g \times A_g' \times A_g \times S_g$.

Likewise, we use symbols to denote a real-world ES (M_r), a perfect EM (M_p), and a pragmatic EM (M_g). These are represented as:

$$M_r = E_r' \times E_r \times A_r' \times A_r \times S_r \times U_r$$
$$M_p = E_p' \times E_p \times A_p' \times A_p \times S_p \times U_p$$
$$M_g = E_g' \times E_g \times A_g' \times A_g \times S_g \times U_g$$

These symbols help us distinguish and calculate the various components of ECs and EMs in different contexts.

3.7. Fundamental issues in evaluatology

This subsection presents three fundamental issues in Evaluatology.

3.7.1. Ensure transitivity of EMs

The key to the effectiveness and efficiency of evaluations in different scenarios is to establish a series of EMs that ensure the transitivity of the primary characteristics.

In Section 3.6, we have effectively examined and described a real-world ES, along with its corresponding EM, which we call a perfect EM, and their interconnections.

The real-world ES system presents three notable obstacles. Firstly, the presence of numerous confounding creates a challenge as they cannot be completely eliminated. Secondly, the existence of multiple irrelevant problems or tasks adds another layer of complexity. Lastly, regardless of the nature of the problems or tasks involved, there is a tendency for the system to exhibit bias towards certain clusters exhibited by problem or task instances, algorithmic mechanisms, and their instantiations.

To overcome these obstacles, a perfect EM, which possesses several key characteristics, is proposed. Firstly, the model eliminates any irrelevant problems or tasks that may be used to derive evaluation standards for assessing different subjects. Secondly, the model enables the free setting of an REM. Thirdly, the model allows for a comprehensive

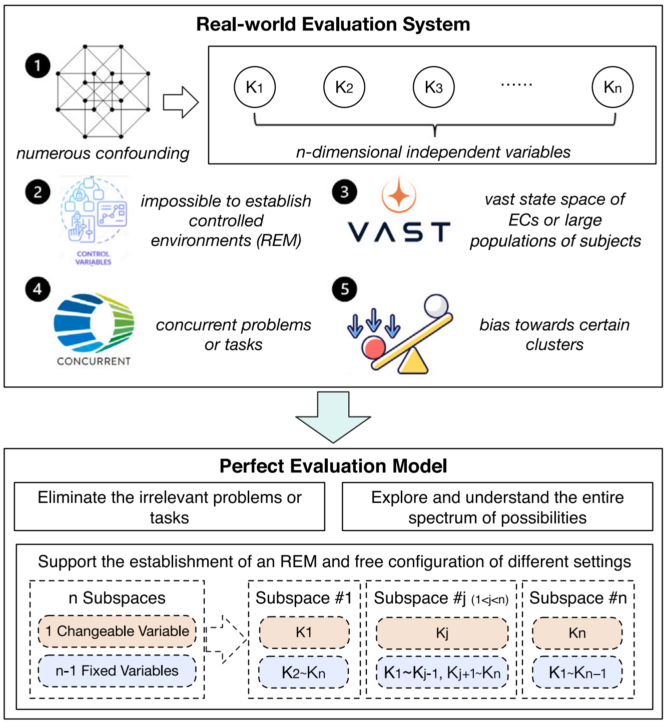

Real-world Evaluation System
1
numerous confounding
K1
K2
K3
Kn
n-dimensional independent variables
2
impossible to establish controlled environments (REM)
3
VAST
vast state space of ECs or large populations of subjects
4
CONCURRENT
concurrent problems or tasks
5
bias towards certain clusters
Perfect Evaluation Model
Eliminate the irrelevant problems or tasks
Explore and understand the entire spectrum of possibilities
Support the establishment of an REM and free configuration of different settings
n Subspaces
1 Changeable Variable
n-1 Fixed Variables
Subspace #1
K1
K2~Kn
Subspace #j (1<j<n)
Kj
K1~Kj-1, Kj+1~Kn
Subspace #n
Kn
K1~Kn−1

Fig. 8. A perfect EM resembles a real-world ES.

exploration and understanding of the entire spectrum of possibilities in terms of problem or task instances, algorithmic mechanisms, and their instantiations.

Compared to a real-world ES $M_r = E'_r \times E_r \times A'_r \times A_r \times S_r \times U_r$, a perfect EM eliminates its irrelevant problems or tasks, represented by $M_p = E'_p \times E_p \times A'_p \times A_p \times S_p \times U_p$. Transforming a real-world ES into a perfect EM should ensure the transitivity of the following characteristics (see Fig. 8).

$$E'_p \subset E'_r$$
$$SE_p/E'_p \supset SE_r/E'_p$$
$$SA'_p/E'_p/E_p \supset SA'_r/E'_p/E_p$$
$$SA_p/E'_p/E_p/A'_p/S_p \supset SA_r/E'_p/E_p/A'_p/S_p.$$

The perfect EM encompasses large populations of problem or task instances, algorithm-like mechanisms, their instantiations, and support systems. Also, it entails a vast number of independent variables.

To overcome this difficulty, it becomes essential to propose a pragmatic EM that simplifies the perfect EM in two ways: (1) reducing the number of independent variables and (2) sampling from the extensive populations of support systems, problem or task instances, algorithm-like mechanisms, and their instantiations.

A pragmatic EM adopts a sampling approach on the perfect EM, resulting in a smaller space to work with. To formalize this process, we introduce the notation $s(\cdot)$ to represent the sampling function. Additionally, the pragmatic EM streamlines the independent variables within the perfect EM by excluding those that have minimal impact.

For each element m_g in the sampled space M_g, which is a subset of the perfect EM M_p, we denote the corresponding element in M_p as m_p. When transforming a perfect EM into a pragmatic EM, it is essential to maintain the transitivity of the following characteristics:

$M_g = s(M_p)$: The sampled space M_g is obtained through the application of the sampling function s on the perfect EM M_p.

$M_g \subset M_p$: The sampled space M_g is a subset of the perfect EM M_p.

$m_g = (k_1, \dots, k_{n'}) \in M_g$: Each element m_g in the sampled space M_g consists of a set of independent variables $(k_1, \dots, k_{n'})$.

$m_p \in M_p$ is the matched element in the perfect EM M_p corresponding to m_g. m_p consists of a set of independent variables $(k_1, \dots, k_{n''})$. n'' is greater than n', ensuring that the corresponding element in the perfect EM includes at least as many independent variables as that of the element in the pragmatic EM.

3.7.2. Perform cost-efficient evaluation with controlled discrepancies

By disregarding the accuracy of an EM, conducting evaluations solely through the establishment of an REM within a perfect EM may indeed result in maximum confidence. However, this approach also comes with a significant drawback — the exorbitant cost it entails. The process of creating an REM within a perfect EM can be prohibitively expensive, making it impractical for many organizations. Therefore, it is crucial to strike a balance between ensuring the discrepancy threshold of the evaluation outcomes and managing the associated costs when implementing evaluation processes.

When creating a pragmatic EM from a perfect EM, the discrepancy threshold ϵ, which is a discrepancy limit that can be tolerated in an evaluation scenario, holds the potential to exert a profound influence on the evaluation results and, in certain instances, it would give rise to grave concerns, particularly in the context of safety-critical tasks where failure could lead to detrimental side effects such as harm, loss of life, or significant environmental damage. So, after thoroughly understanding the stakeholders' evaluation requirements, a risk function $\gamma(\cdot)$ could be predefined. When the stakes are high, and there is a greater risk associated with the evaluation outcomes, it becomes imperative to have a lower discrepancy threshold between the evaluation outcomes of a pragmatic EM and a perfect EM. This is because the potential consequences of making a wrong decision or drawing inaccurate conclusions become more significant.

In Section 3.7.1, we use the notation $s(\cdot)$ to represent the sampling function. In creating a pragmatic EM from a perfect EM, the accuracy of EM decreases. We use the notation $m(\cdot)$ to represent this modeling process. We use the notation $e(\cdot)$ to represent the process of ensuring different equivalency levels of EC.

We introduce a discrepancy function of the evaluation outcomes $\mathrm{disc}(\cdot)$ between M_g and M_p. When the discrepancy is 0, it indicates that M_g and M_p are equivalent.

The discrepancy function of the evaluation outcomes $\mathrm{disc}(\cdot)$ between M_g and M_p is defined as follows. In the formulation, $\rho(\cdot)$ is a measurement function, and $v(\cdot)$ is a value function. Besides, we define the evaluation cost as the product of a constant μ and the space capacity of M_g. This cost factor allows us to incorporate the resource constraints and practical considerations associated with the evaluation process.

$$\begin{cases} \text{discrepancy threshold } \epsilon = \gamma(\cdot), \\ \mathrm{disc(M_g, M_P)} = \mathrm{disc}(v(\rho(e(m(s(M_p)))))), v(\rho((M_p))), \\ \mathrm{cost(M_g)} = \mu \| M_g \|. \end{cases}$$

Based on the above formulation, the evaluation issue of balancing evaluation cost and the discrepancies in the evaluation outcomes can be framed as an optimization problem. The objective is to minimize the evaluation cost, represented by cost while ensuring the discrepancies in the evaluation outcomes, denoted as $\mathrm{disc}(M_g, M_p)$, do not exceed a predefined discrepancy threshold ϵ (see Fig. 9).

The optimization problem can be formulated as follows:

$$\arg\min \mathrm{cost(M_g)} \text{ subject to } (\mathrm{disc}(M_g, M_p) < \epsilon).$$

3.7.3. Ensure evaluation traceability

According to the third axiom of evaluation, for a well-defined subject, the divergence in the evaluation outcomes can be attributed to disparities in ECs, thereby establishing evaluation traceability.

Conceptually, traceability asks for a quantified mapping between the differences in the input and output of the value function $v(\cdot)$ decided by the evaluation community and the measurement function $\rho(\cdot)$

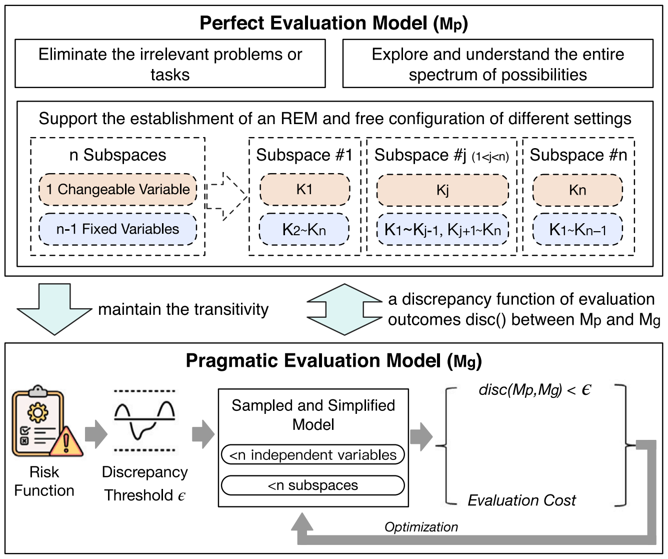

Perfect Evaluation Model (Mp)
Eliminate the irrelevant problems or tasks
Explore and understand the entire spectrum of possibilities
Support the establishment of an REM and free configuration of different settings
n Subspaces
1 Changeable Variable
n-1 Fixed Variables
Subspace #1
K1
K2~Kn
Subspace #j (1<j<n)
Kj
K1~Kj-1, Kj+1~Kn
Subspace #n
Kn
K1~Kn-1
maintain the transitivity
a discrepancy function of evaluation outcomes disc() between Mp and Mg
Pragmatic Evaluation Model (Mg)
Risk Function
Discrepancy Threshold ϵ
Sampled and Simplified Model
<n independent variables
<n subspaces
disc(Mp,Mg) < ϵ
Evaluation Cost
Optimization

Fig. 9. Proposing a pragmatic EM based on the evaluation risk function.

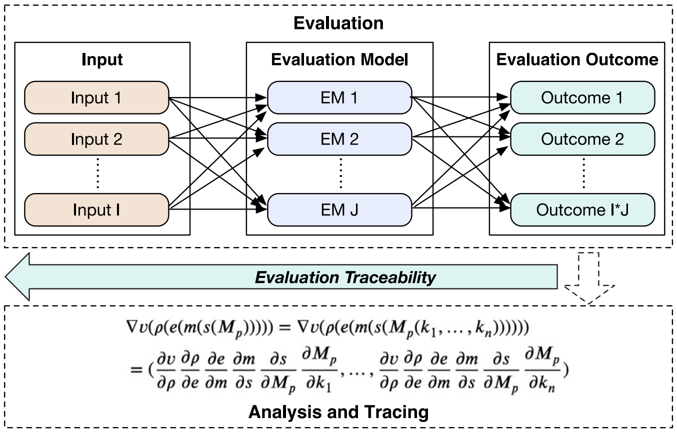

Evaluation
Input
Input 1
Input 2
Input I
Evaluation Model
EM 1
EM 2
EM J
Evaluation Outcome
Outcome 1
Outcome 2
Outcome I*J
Evaluation Traceability
$\nabla v(\rho(e(m(s(M_p))))) = \nabla v(\rho(e(m(s(M_p(k_1,\dots,k_n))))))$
$= (\frac{\partial v}{\partial \rho}\frac{\partial \rho}{\partial e}\frac{\partial e}{\partial m}\frac{\partial m}{\partial s}\frac{\partial s}{\partial M_p}\frac{\partial M_p}{\partial k_1},\dots,\frac{\partial v}{\partial \rho}\frac{\partial \rho}{\partial e}\frac{\partial e}{\partial m}\frac{\partial m}{\partial s}\frac{\partial s}{\partial M_p}\frac{\partial M_p}{\partial k_n})$
Analysis and Tracing

Fig. 10. Evaluation traceability.

through the evaluation process, described by the mathematical model we formulated above, and the model in the following formulation is a perfect EM. We discover that this concept aligns well with the mathematical notation of the gradients of a function, which gives the rate of changes in the output for each input variable. Fig. 10 shows how to ensure the evaluation traceability. In the context of evaluation, the gradient of evaluation outcomes can be written as follows, which is a matrix or tensor:

$$\nabla v(\rho(e(m(s(M_p))))) = \nabla v(\rho(e(m(s(M_p(k_1,\dots,k_n))))))$$
$$= (\frac{\partial v}{\partial \rho}\frac{\partial \rho}{\partial e}\frac{\partial e}{\partial m}\frac{\partial m}{\partial s}\frac{\partial s}{\partial M_p}\frac{\partial M_p}{\partial k_1},\dots,\frac{\partial v}{\partial \rho}\frac{\partial \rho}{\partial e}\frac{\partial e}{\partial m}\frac{\partial m}{\partial s}\frac{\partial s}{\partial M_p}\frac{\partial M_p}{\partial k_n}).$$

The closed-form mathematical expression is not always available for various EC components in evaluation. Nevertheless, we can follow the method of acquiring gradients in numerical methods by creating perturbations in the ECs for various input variables and observing the differences in the composite evaluation outcomes, thus approximating the gradients.

3.7.4. Connect and correlate evaluation standards across diverse disciplines

While the constituents comprising ECs may differ across distinct evaluation scenarios, a governing fundamental component of ECs emerges as the evaluation standard. The shared qualities of evaluation standards, as denoted by the SDE characteristics, indicate the potential for establishing a correlation between evaluation standards across diverse disciplines.

The evaluation standard serves as a fundamental pillar within any evaluation model. By establishing connections between evaluation standards across various disciplines, we have the potential to construct a comprehensive framework encompassing evaluation issues in all fields. This holistic framework, known as the grand unified theory of evaluatology, allows for a thorough exploration of evaluation-related matters.

Before defining an evaluation standard, it is essential to understand the nature of the stakeholders' primary problems or tasks. In the rest of this article, a problem or task refers explicitly to a computational problem. Make sure to distinguish between a problem and a problem instance: A problem is an infinite collection of problem instances, each of which is a problem with concrete configurations, which we have elaborated in Section 3.5.1.

Computational complexity theory provides the basis for understanding the nature of primary problems. For example, complexity classes – that are defined by bounding the time or space used by the algorithm – can be used to understand the problem's nature [21]. Computability theory [22] seeks a more general question about all possible algorithms that could be used to solve the same problem. That is to say, the computability theory provides a viable solution to answer whether a problem is solvable. In understanding the problem instance, the theory of analysis of algorithms [23] can be used to analyze the amount of resources needed by a particular algorithm to solve a problem instance.

While the computational complexity and other theories mentioned above lay the foundation, the formulation of evaluation standards introduces novel concerns. The first challenge lies in addressing scenarios where articulating a mathematical model explicitly becomes an insurmountable task. Regrettably, this circumstance is not uncommon, as numerous problems defy expression through a mathematical model at present.

To ensure the definitiveness and equivalency of the evaluation standard, it is imperative to establish a rigorous problem space definition and a problem instance space definition, which provides the quantitative foundation for the comparability and traceability of different EMs.

The representativeness of the evaluation standard is a crucial aspect that warrants discussion. Understanding the composition of problems is crucial in identifying the problem or task that best represents the whole. For instance, across various scientific and engineering disciplines, problems often exhibit a hierarchical structure, where a significant problem can be broken down into several smaller problems commonly referred to as "dwarfs" [24]. This pattern can be considered one of the foundational structures within problem domains. Gaining profound insights into the structural aspects of problems proves immensely valuable when assessing complex and multifaceted subjects.

Fig. 11 shows how to correlate evaluation standards across diverse disciplines. In an optimal scenario, we can identify evaluation standards that embody the SDE characteristics across various disciplines. Ultimately, we can establish connections and correlations between evaluation standards from different fields, giving rise to the grand unified theory of evaluatology. The objective of this theory is to present a hierarchical framework of evaluation standards. Within this framework, we can identify a minimal set of fundamental evaluation standard "dwarfs" along with their respective variations. Complex evaluation standards are formed by combining two or more of these evaluation standard dwarfs and their variants. This hierarchical structure of evaluation standards will greatly facilitate the reuse and sharing of knowledge.

4. Benchmarkology: the engineering of evaluation

This section unveils the core essence of a benchmark and introduces the benchmark-based engineering of evaluation, which we call benchmarkology. Furthermore, we provide guidelines and workflows within the realm of benchmarkology.

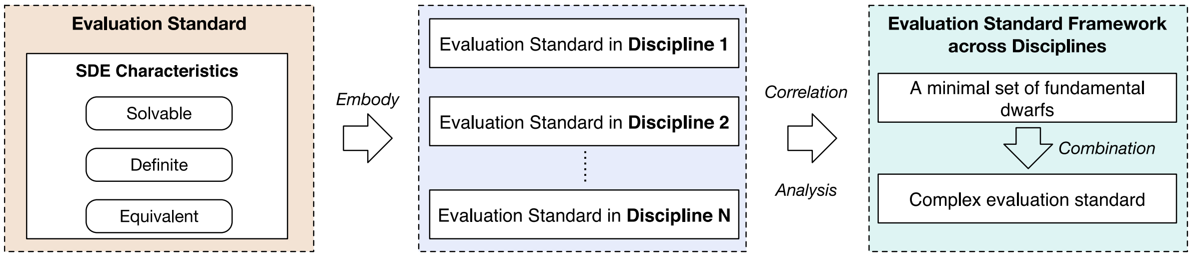

Evaluation Standard
SDE Characteristics
Solvable
Definite
Equivalent
Embody
Evaluation Standard in Discipline 1
Evaluation Standard in Discipline 2
Evaluation Standard in Discipline N
Correlation
Analysis
Evaluation Standard Framework across Disciplines
A minimal set of fundamental dwarfs
Combination
Complex evaluation standard

Fig. 11. Correlating evaluation standards across diverse disciplines.

4.1. The essence of a benchmark

Benchmarks are extensively employed across various disciplines, albeit lacking a formal definition. Based on the science of evaluation, we propose a precise delineation of a benchmark as *a simplified and sampled EC, specifically a pragmatic EC, that ensures different levels of equivalency, ranging from LEECs to EECs.*

Within the framework of this definition, a benchmark comprises three essential constituents. The first constituent is the *stakeholder's evaluation requirements*, which encompass various factors. These include the risk function, which evaluates the potential risks associated with the benchmark. Additionally, the discrepancy threshold of the evaluation outcomes, which determines the acceptable level of deviation in evaluation outcomes, is considered. The evaluation confidence interval plays a crucial role in predicting the parameter of a perfect EM. Lastly, the evaluation cost of EM is taken into account, and the resources required for conducting the evaluation are assessed. By considering these elements, the benchmark can effectively address the evaluation requirements of stakeholders.

The second constituent of the benchmark framework is the *EC configuration and mechanisms*. This includes several elements crucial for the benchmark's effectiveness. Firstly, it involves defining the set of problems or tasks that the stakeholders face when addressing them. Additionally, it encompasses the set of equivalent problem or task instances, which helps ensure specificity in the evaluation process. The benchmark also considers algorithm-like mechanisms and their instantiations, which play a significant role in solving the defined problems or tasks. The support systems, which provide necessary resources and environments, are also taken into account.

Moreover, the benchmark provides the means to eliminate confounding variables that may affect the evaluation outcomes. Also, the benchmark provides the mechanism to ensure varying levels of EC equivalency, determining the extent to which different benchmark instances can be considered equivalent.

By considering these EC configurations and mechanisms, the benchmark can provide a comprehensive and standardized approach to evaluating problems or tasks.

The third constituent is the *metrics and reference*, including the definitions of quantities, the value function, composite evaluation metrics, the reference subject, and the reference evaluation outcomes.

In the subsequent sections of this article, we will refer to these three constituents as the complete constituents of a benchmark. Fig. 12 shows the three essential constituents of a benchmark.

4.2. The goal of benchmarkology

As expounded upon in Section 3, the science and engineering of evaluation, known as evaluatology, aims to apply the EECs to various subjects and establish an REM. A benchmark can be viewed as a simplified and sampled EC, specifically a pragmatic EC, that ensures different levels of equivalency, ranging from LEECs to EECs. In this context, a benchmark-based approach to the evaluation problem is considered a feasible engineering methodology, given the widespread use of benchmarks across various disciplines. Consequently, we propose the formal definition of *benchmarkology* as an engineering discipline concerned with the quantitative assessment of diverse subjects using benchmarks.

Undoubtedly, the theory of evaluatology serves as the foundation for benchmarkology. Nevertheless, benchmarkology has its unique objective — to furnish guiding principles and engineering evaluation methodologies.

4.3. The principles in building benchmarks

In Section 3, we have extensively discussed the fundamental axioms of evaluatology. Nonetheless, this particular subsection delves deeper into the principles that underpin the creation of benchmarks derived from these four evaluation axioms.

The first principle focuses on the validity of metrics within a benchmark. According to the First Axiom of Evaluation, also known as the Axiom of the Essence of Composite Evaluation Metric, there are three criteria to determine the validity of metrics in benchmarks. If a metric does not meet these criteria, it is considered invalid. The three criteria are as follows:

1. The metric should be a base quantity.
2. The metric should represent another quantity that has inherent physical significance.
3. The metric should be a composite evaluation metric that is explicitly defined by a value function.

The second principle pertains to the comprehensiveness of the configurations of a valid benchmark. According to the Second Axiom of Evaluation, known as the Axiom of True Evaluation Outcomes, when a well-defined subject is equipped with a well-defined EC, its evaluation outcomes possess true values. A well-defined EC should reveal all its well-defined components. Each component within the EC plays a critical role in determining the evaluation outcomes. When the components of the EC are not well-defined, uncertainty is introduced into the evaluation process. Without clear and specific components, the evaluation outcomes become unpredictable and lack reliability.

The evaluation outcome using a benchmark should reveal the complete evaluation configurations. Unfortunately, many contemporary benchmarks, both in terms of state-of-the-art and state-of-the-practice, have failed to disclose their comprehensive evaluation configurations fully.

Some benchmarks, like the widely-used CPU benchmark SPECCPU, may omit certain constituents or their components, e.g., the support system. In such cases, it becomes essential to clearly define the conditions under which the simplification is made, ensuring that the benchmark can still provide meaningful and valid results. By providing these detailed evaluation configurations, we can ensure that the benchmark remains a reliable tool for evaluation purposes.

The third principle centers around the concept of benchmark traceability. In accordance with the Third Axiom of Evaluation, also known as the Axiom of Evaluation Traceability, it is crucial to establish benchmark traceability to enable the comparison of different benchmarks. This means that it is a top priority to trace the discrepancies

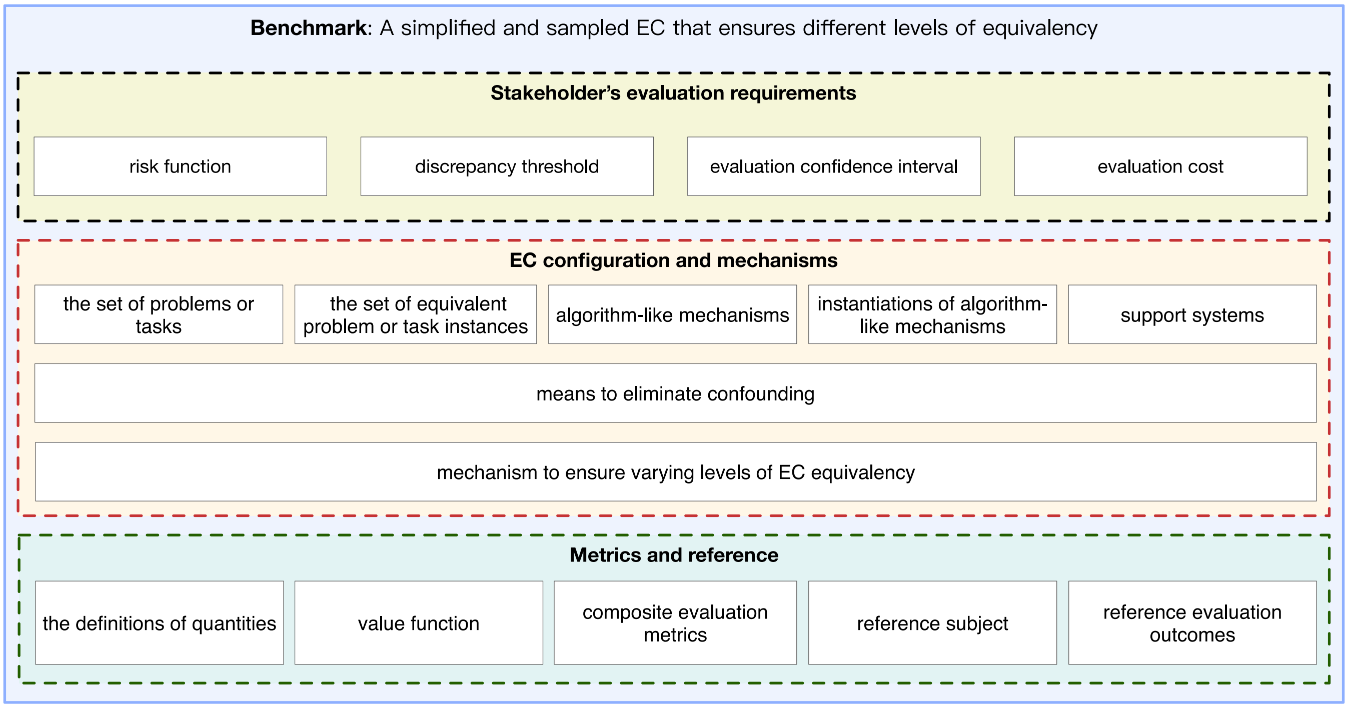

Benchmark: A simplified and sampled EC that ensures different levels of equivalency
Stakeholder's evaluation requirements
risk function
discrepancy threshold
evaluation confidence interval
evaluation cost
EC configuration and mechanisms
the set of problems or tasks
the set of equivalent problem or task instances
algorithm-like mechanisms
instantiations of algorithm-like mechanisms
support systems
means to eliminate confounding
mechanism to ensure varying levels of EC equivalency
Metrics and reference
the definitions of quantities
value function
composite evaluation metrics
reference subject
reference evaluation outcomes

Fig. 12. A benchmark comprises three essential constituents.

in evaluation outcomes back to variations in the configurations of different benchmarks.

The fourth principle encompasses the validity of the speedup of two evaluations obtained using the benchmark. It emphasizes the importance of understanding the implications of the speedup obtained in different pragmatic EMs in relation to those of the real-world ES or perfect EM.

Comparison is a common practice using benchmarks. In practice, the speedup is often a key metric of interest that is obtained in different pragmatic EMs. When we compare two evaluations, we use the benchmarks as a means to infer the true speedup in the real-world ES or pefect EM.

To illustrate this principle, let us consider a specific example where we compare two subjects, Subjects A and B, and obtain a speedup (either greater than or less than 1).

In a simulated CPU scenario (a pragmatic EM), the reported speedup is 1.3, indicating an improvement in performance for Subject A compared to Subject B. However, the ratio between the speedup in the EM and the corresponding speedup in the real-world ES has a 90% confidence interval of [0.7, 1.9]. This means that the ratio could be any value within this interval, including 1.6.

If the ratio of speedup is indeed 1.6, the actual speedup in the real-world ES would be 1.3 divided by 1.6, resulting in a value of 0.8. This indicates a degraded performance for Subject A in comparison to Subject B, reported on the real-world ES, which is contradicted by an improvement in performance reported on a simulated system.

It is crucial to note that relying solely on the reported speedup in the EM without considering its implication in the real-world ES can lead to misleading interpretations and decisions. To ensure the validity of evaluation outcomes, it is essential to take into account the confidence interval associated with the speedup ratio between the EMs and real-world ES.

4.4. The universal methodology in benchmarkology

The aforementioned principles offer valuable insights into the fundamental components of a benchmarkology workflow, as shown in Fig. 13.

4.4.1. Understand stakeholders' evaluation requirements

During the initial phase, it is essential for evaluators to gain a comprehensive understanding of the stakeholders' evaluation requirements, the first constituent of the benchmark, which we discussed in detail in Section 4.1. This crucial step allows for the alignment of these requirements with the overall purpose of the evaluation.

During this phase, a thorough examination of the evaluation risk function, the discrepancy threshold of evaluation outcomes, and the evaluation cost should be conducted. Additionally, for quantities or variables of interest, it is crucial to establish their evaluation confidence interval when using the benchmark. This quantification allows for an assessment of how effectively the benchmark can infer or predict the parameters of a perfect EM.

Unfortunately, in state-of-the-art or state-of-the-practice benchmarks, the importance of this phase is often overlooked. There are two possible reasons for this oversight.

Firstly, in certain evaluation scenarios, the discrepancy in evaluation outcomes, whether intermediate or large, may not have significant consequences. However, it is crucial to note that this is not the case in scenarios involving safety-critical, mission-critical, and business-critical applications. In these situations, even minor deviations can have severe impacts on the overall outcome.

Secondly, the benchmark process itself is an engineering practice that emphasizes iterative and refined operation. As a result, it implicitly incorporates some procedures of this phase.

Therefore, it is imperative to recognize the significance of understanding stakeholders' evaluation requirements, particularly in scenarios where the stakes are high and any discrepancy from expected outcomes can have critical implications.

4.4.2. Design and implement intricate evaluation mechanisms and policies

This phase plays a crucial role, particularly in complex evaluation scenarios, and can be quite costly. Its primary aim is to provide a solid foundation for generating a benchmark.

The real-world ES reflects the complexities and nuances of actual evaluation environments. In this phase, the evaluator takes on the crucial task of building and investigating the real-world ES. The methodology discussed in Section 3.6 can serve as a helpful guide for this process.

During the investigation, evaluators need to consider several aspects carefully. One aspect involves identifying and eliminating irrelevant problems or tasks that may not be applicable to the assessment of the subjects under evaluation. This ensures that the evaluation focuses on relevant and meaningful aspects of the subject. Another important consideration is that evaluators must recognize any constraints or factors that may impact the evaluation process. This understanding lays the groundwork for creating a perfect EM that explores all the possibilities. A perfect EM serves as an ideal evaluation model, and the design and implementation of this model are also key focus in this phase. We have discussed its main concerns in Section 3.6.

Additionally, in this phase, two key policies and their accompanying procedures are of utmost importance. Firstly, the modeling policy and procedure guide the process of transforming the real-world ES into an EM, striking a balance between accuracy and cost. This involves capturing the essential elements and characteristics of the real-world ES in the model while ensuring that the modeling process is efficient and cost-effective.

Secondly, the sampling policy and procedure play a vital role in transitioning from a perfect EC to a pragmatic EC. This transition aims to save on evaluation costs while still maintaining a high level of confidence in the evaluation results. The sampling policy and procedure guide the selection of representative samples from the perfect EC, ensuring that the pragmatic EC captures the essential aspects and characteristics of the perfect EC while being more practical and resource-efficient.

By following these policies and procedures, the evaluation process is adapted to real-world conditions and constraints. The modeling policy and procedure enable the creation of an EM that accurately represents the real-world ES, while the sampling policy and procedure ensure that the pragmatic EC reflects the essential elements of the perfect EC. This allows for a more effective and efficient evaluation process that balances accuracy, cost, and confidence.

Overall, with the facilitation of the study of real-world ES and the perfect EM, these modeling and sampling policies and procedures are essential in this phase to guide the modeling and sampling processes, ensuring that the evaluation process is well-suited to real-world conditions and constraints.

4.4.3. *Decide representative evaluation standards*

The third phase is to decide the representative evaluation standards that guarantee the least EECs, ensuring the comparability of the evaluation outcomes. The main objective of this phase is to carefully consider the relevant stakeholders involved and gain a deep understanding of their principal interests and concerns. This phase requires evaluators to identify and comprehend the primary problems or tasks that need to be addressed.

It is important to note that each stakeholder has a unique perspective, leading to subtle differences in the problems or tasks they face. Therefore, it is crucial for evaluators to recognize and take into account these varying perspectives, ensuring that the evaluation standards are comprehensive and reflective of the diverse interests of the stakeholders involved.

In Section 3.5.4, we have explored the concept of evaluation standards and how they are derived from the abstract problem or task at hand. Building upon this understanding, the evaluator's next step is to determine the specific evaluation standards by selecting representative instances of the primary problems or tasks faced by the stakeholders.

It is important to recognize that different stakeholders will have distinct evaluation standards. This is because their perspectives and priorities vary based on their unique roles and interests. As a result, it is essential for evaluators to consider these differences and tailor the evaluation standards accordingly to ensure that they capture the specific needs and concerns of each stakeholder involved.

It is true that previous evaluation and benchmark practices have often lacked consistent discussions on what qualifies as an evaluation standard. However, we believe that our universal definition of evaluation standards has the potential to be applicable across various evaluation scenarios in different disciplines. Our aim is to provide a comprehensive framework that can guide evaluators in establishing effective evaluation standards.

Furthermore, we acknowledge the importance of recognizing that a realistic benchmark can only capture a small sample of huge populations of instances that are derived from a large population of problems or tasks. Despite the challenges that may arise when explicitly stating the problem or task, we remain committed to adopting a systematic approach to our thinking. This allows us to navigate through such complexities and develop meaningful evaluation standards that align with the objectives of the evaluation process.

4.4.4. *Design and implement ECs with different levels of equivalency*

Based on the outputs from Phases Two and Three, the subsequent stage involves the design and implementation of ECs with different levels of equivalency. This task varies from different subjects. In general, this phase needs to consider algorithm-like mechanisms and their instantiations, as well as the support systems.

Furthermore, in this phase, it is essential to address the levels of EC equivalency. This involves determining the extent to which different benchmark instances can be considered equivalent. It requires careful consideration of which components can be disregarded or simplified in order to streamline the benchmark process while maintaining its validity and reliability.

Additionally, the benchmark needs to establish mechanisms to eliminate confounding that may impact the evaluation outcomes. Confounding variables can introduce biases or distortions into the evaluation results, affecting their accuracy and reliability. One approach to address confounding is by employing our proposed REM methodology. This methodology provides a systematic framework to identify and eliminate confounding variables and ensure that the evaluation outcomes are not influenced by extraneous factors.

4.4.5. *Perform measurement and/or testing*

The fourth phase encompasses measurements and/or testing, guided by the principles and practices of the metrology and testing theory. The measurement and testing process serves multiple purposes. Firstly, evaluators must determine which properties or quantities to measure, keeping in mind what base quantities and other quantities that carry physical meaning are.

Furthermore, evaluators must also consider the cost of measurement and testing, ensuring that it aligns with budgetary constraints. It becomes imperative for them to make informed decisions on various aspects, such as how, when, and to what extent to perform testing, sampling, and measuring these properties or quantities. By doing so, they can effectively manage resources while still obtaining valuable data for their research.

4.4.6. *Perform assessments*

The fifth phase entails assessment, wherein the defining of a value function takes precedence. The rationale behind establishing a value function lies in the aim of encapsulating numerous quantities that surpass our capacity for recognition into a singular metric. This value function serves as a proposed function, mapping the target properties or quantities measured during the preceding phase to the evaluation outcomes in order to reflect the concerns or interests of the stakeholders. Given that stakeholders often possess varying concerns or interests, it is common to propose multiple value functions from different perspectives. Different communities may reach a consensus on how to define a value function. Generally, evaluators must engage in consultation with the stakeholders to define a composite evaluation metric in the form of

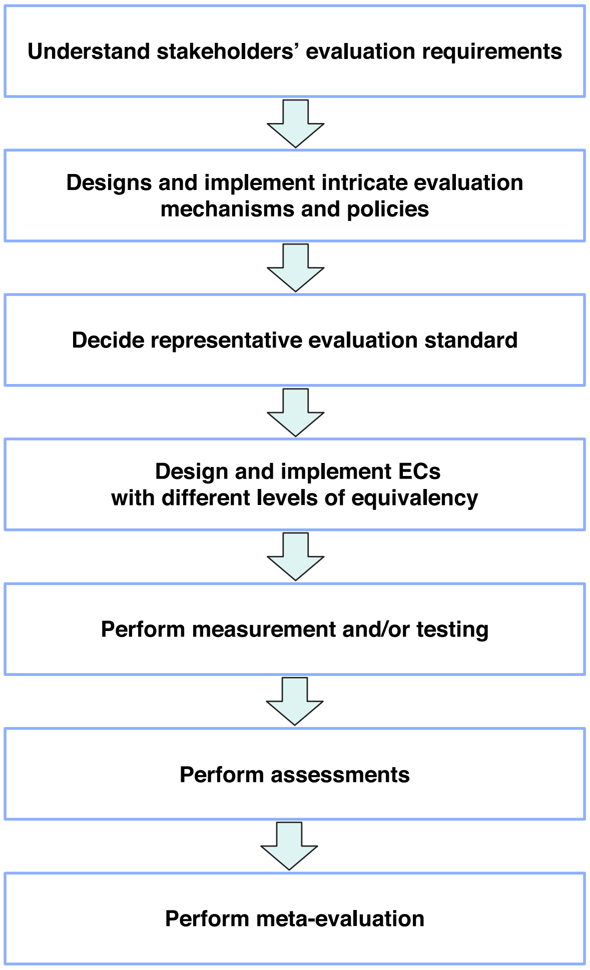

Understand stakeholders' evaluation requirements
Designs and implement intricate evaluation mechanisms and policies
Decide representative evaluation standard
Design and implement ECs with different levels of equivalency
Perform measurement and/or testing
Perform assessments
Perform meta-evaluation

Fig. 13. The universal methodology in benchmarkology.

a value function that truly encapsulates the stakeholders' concerns or interests.

Subsequently, evaluators can compare the obtained evaluation outcomes to the reference evaluation outcomes, ultimately enabling judgment on different subjects, such as performance, value, merit, weaknesses, worth or significance, as well as positive or negative effects.

4.4.7. Perform meta-evaluation

The last phase involves conducting meta-evaluations. The evaluators are tasked with reviewing all the evaluation processes and determining whether the theory, data, or evidence produced can substantiate the conclusions drawn in the evaluations. In this phase of meta-evaluation, various perspectives are taken into account, including the dimensionality of measurement data [7], the reliability of measurement results, the validity of evaluation outcomes, the traceability of benchmarks, as well as the cost and cost-efficiency of the evaluation itself [1].

The "dimensionality" of measurement data refers to the number and nature of variables that are reflected in the assessment [7]. The reliability of measurement results pertains to the extent to which the measured values accurately reflect the true values. The validity of the evaluation outcome denotes the degree to which the statistics of the benchmark can infer the parameter of the real-world evaluation setting or a perfect evaluation model in terms of the metrics of confidence level and confidence interval.

Two approaches can be undertaken to manage benchmark traceability. Firstly, it is essential to develop a comprehensive mathematical model capable of capturing the influence of the discrepancies of the evaluation configurations on the evaluation results. This model will serve as a foundation for interpreting and comprehending the evaluation outcomes. Additionally, the community must engage in continual benchmark comparisons. Drawing inspiration from the calibration practices in the field of metrology, this consistent comparison and alignment of different benchmarks will ensure consistency and accuracy in evaluation procedures.

Constrained by the limitations of the budget, the aforementioned six phases can be carried out iteratively, employing a trial-and-error methodology.

5. Why it is essential to develop evaluatology?

In this section, we will begin by conducting a comprehensive review of the state-of-the-art and state-of-the-practice evaluation methods and benchmarks. This review will provide us with a solid understanding of the current landscape and help identify areas where advancements are needed.

To illustrate the importance of advancing the science and engineering of evaluation, we will focus on the evaluation of CPU performance as a prime example. Meanwhile, we will critically reflect on the existing state-of-the-art and state-of-the-practice evaluation methods and benchmarks. This reflection will enable us to identify any limitations or gaps that need to be addressed for more accurate and meaningful evaluations.

Furthermore, we will explore the advantages that the field of evaluatology brings to the table. To ensure clarity and understanding, we will provide a concise summary of the distinctions between evaluation, measurement, and testing.

To ensure consistency and alignment with our proposed universal terminology, we will utilize our established terminology framework when discussing state-of-the-art and state-of-the-practice evaluation and benchmark cases.

5.1. Evaluations across different academic fields

This subsection presents a concise overview of the cutting-edge evaluations in a range of academic disciplines, as well as the prevailing evaluation practices.

5.1.1. Observation study methodologies

Observational study methodologies are widely used in the fields of business science, finance, and education. Even based on a random sample, an observational study still falls short of effectively revealing the cause-and-effect relationships.

Evaluations in the field of business science:

Camp [8] defines benchmarking as "the search for those best practices that will lead to the superior performance of a company". Benchmarking consists of two primary steps [8]: (1) establishes operation targets based on industry best practices; (2) "a positive, proactive, structured process leads to changing operations and eventually achieving superior performance and competitive advantage". In the study conducted by Andersen et al. [25], the essence of benchmarking is summarized as the quest for knowledge and learning from others.

Evaluations in the fields of finance and education:

In the fields of finance and education, indices are widely used as benchmarks to assess the overall performance of the individuals or systems under study. These indices are derived by calculating the weighted average of a selected group of individuals or systems [9].

For example, stock market indices are used as benchmarks to assess the stock market's performance in the finance field. These indices are derived by calculating the weighted average of a selected group of representative stocks [9]. Some widely recognized stock market indices include the Dow Jones Industrial Average, the S&P 500, the NASDAQ

Table 1
The base quantity, value function, and the reference machine specified in different CPU benchmark suites in Section 5.

Benchmark	Category	Base quantity	Value function	Reference machine
SPEC CPU2006	SPECspeed	Execution time	x_i = time on a reference machine/time on the evaluation machine [27]; $\bar{x} = \sqrt[n]{\prod_i^n x_i}$	The Ultra Enterprise 2 with 296 MHz UltraSPARC II chips [28]
	SPECrate	Execution time	x_i = number of copies * (time on a reference machine/time on the evaluation machine) [27]; $\bar{x} = \sqrt[n]{\prod_i^n x_i}$	
SPEC CPU2017	SPECspeed	Execution time	x_i = time on a reference machine/time on the evaluation machine [27]; $\bar{x} = \sqrt[n]{\prod_i^n x_i}$	The Sun Fire V490 with 2100 MHz UltraSPARC-IV+ chips [27]
	SPECrate	Execution time	x_i = number of copies * (time on a reference machine/time on the evaluation machine) [27]; $\bar{x} = \sqrt[n]{\prod_i^n x_i}$	
PARSEC		Execution time	x_i = time on the evaluation machine	—

Composite, and the Shanghai Stock Exchange Composite Index. Different indices employ varying calculation methods. The most common approach is the weighted average method, which determines the index value based on the weighted average of the constituent stock prices. Another method is the geometric mean method, which calculates the geometric average of the stock prices and adjusts it using a base period price. Typically, stock market indices are published at the close of each trading day. Some index providers offer real-time index data, enabling investors to stay informed about the latest market conditions.

The Brent benchmark is used to determine the price of Brent crude oil [26]. Brent crude oil is a type of light and low-sulfur crude oil produced from oil fields in the North Sea region. Due to its relatively stable supply and high quality, Brent crude oil has become a significant benchmark in the international oil market. Traders, investors, and industry participants worldwide reference the Brent benchmark to track and evaluate the price of Brent crude oil.

In the finance discipline, indexes or benchmarks serve as reference measurements or evaluation results. However, these practices often prioritize data collection and processing over building a solid evaluation theory framework.

5.1.2. *Experimental methodologies*

Experimental methodologies are widely used in the fields of social sciences, computer sciences, psychology, and medicine.

Evaluations in the field of social sciences:

According to Rossi et al. [1], at the earliest, Thomas Hobbes and his contemporaries tried to "use numerical measures to assess social conditions and identify the cause of mortality, morbidity, and social disorganization in the discipline of social science".

Rossi et al. [1] define program evaluation as the process of using social research methods to systematically assess programs aimed at "improving social conditions and our individual and collective well-being", with the goal of providing answers to the stakeholders. Rossi et al. [1] summarize the five domains of evaluation questions and methods that exhibit strong interplays: (1) the need for the programs, (2) program theory and design, (3) program process, (4) program impacts, and (5) program efficiency.

Evaluations in the Field of Computer Science:

The SPEC CPU benchmark suite, known as SPEC CPU [29], is widely recognized as the most renowned benchmark suite for CPU performance evaluation. Throughout its history, six versions of the SPEC CPU benchmark suite have been released, with the latest version being SPEC CPU2017, which can be found in Table 1. The SPEC CPU workloads cover a broad range of CPU-intensive tasks.

The performance evaluation metric used in SPECCPU is based on the execution time. The reported score of SPECCPU represents the ratio of its execution time compared to that of a reference machine. The specific details of a reference machine can be found in Table 1. To ensure the credibility of the results, the overall metrics are calculated as the geometric mean of each respective ratio. Each ratio is based on the median execution time from three runs or the slower of the two runs.

Dongarra et al. [30] proposed the LINPACK benchmark for evaluating high-performance computing (HPC) systems. The LINPACK Benchmark is designed to solve dense linear systems of equations of order n, represented by the equation $Ax = b$. It originated from the development of the LINPACK software package in the 1970s.

The LINPACK benchmark is commonly used to evaluate HPC systems, and the measurement metric is the number of floating-point operations per second (FLOPS). FLOPS represents the count of floating-point operations (FLOPs) performed by the solving algorithm of the LINPACK benchmark, which is calculated as $(2 * n^3/3 + 2 * n^2)$ operations divided by the execution time of the benchmark.

As shown in Fig. 14, ImageNet is a significant benchmark in the field of computer vision, consisting of 14,197,122 high-resolution images manually annotated across 21,841 distinct categories, commonly known as ImageNet-21K [31]. These categories encompass a wide range of objects, animals, and scenes. The ILSVRC (ImageNet Large Scale Visual Recognition Challenge) is an annual computer vision competition that focuses on a subset of ImageNet-21K called ImageNet-1K [32]. It aims to evaluate the performance of deep learning models in tasks such as image classification and object detection, providing specific task configurations and evaluation criteria. ImageNet-1K is primarily used for image classification tasks and consists of 1,281,167 training images, 50,000 validation images, and 100,000 test images. The evaluation metrics commonly used in ILSVRC include Top-1 accuracy, which measures the match between the predicted category and the true category of the image, and Top-5 accuracy, which indicates if the true category of the image is among the top five predicted categories by the model.

Evaluations in the field of medicine:

The evaluation in the field of medicine can be traced back to the early medical eras, although there are no documented records. A rigorous modern medical evaluation methodology and system were established as early as 1938 [34]. Clinical trials, with a history spanning over 250 years, are the primary and widely recognized method for medical evaluation. They are defined as experimental designs to evaluate the potential impact of medical interventions on human subjects [35].

Currently, clinical trials based on experimental designs can be categorized into various types, including randomized trials, double-blind trials, prospective trials, and retrospective trials [36].

As illustrated in Fig. 15, Randomized Controlled Trials (RCTs), considered the gold standard for medical evaluation, possess a rigorous and reliable theoretical framework [37]. However, their high

Table 2
The evaluation outcomes of the Intel Xeon Gold 5120T Processor using different CPU benchmarks show significant discrepancies. SPEC CPU2017 includes two sub-suites: SPECrate and SPECspeed. The results derived from SPECrate and SPECspeed are further categorized into two groups, known as floating-point (FP) and int.

Benchmark	Support platform	Workload	Time (s)	Score	Result	Score meaning
SPECrate	Unix (AIX, HP-UX, Linux, Mac OS X, Solaris), Windows [27]	503.bwaves_r	1483	379.0	96.9 (FP)	Higher scores mean that more work is done per unit of time [27]
		508.namd_r	636	83.7		
		510.parest_r	1742	84.1		
		511.povray_r	1128	116.0		
		519.lbm_r	1420	41.6		
		521.wrf_r	1682	74.6		
		526.blender_r	787	108.0		
		527.cam4_r	998	98.2		
		538.imagick_r	1479	94.2		
		544.nab_r	893	106.0		
		549.fotonik3d_r	2001	109.0		
		554.roms_r	1429	62.3		
		500.perlbench_r	926	96.3	84.3 (INT)	
		502.gcc_r	758	105.0		
		505.mcf_r	1059	85.5		
		520.omnetpp_r	1217	60.4		
		523.xalancbmk_r	786	75.3		
		525.x264_r	1179	83.2		
		531.deepsjeng_r	715	89.8		
		541.leela_r	1197	77.5		
		548.exchange2_r	1338	110.0		
		557.xz_r	824	73.4		
SPECspeed	Unix (AIX, HP-UX, Linux, Mac OS X, Solaris), Windows [27]	603.bwaves_s	224	263.0	48.7 (FP)	Higher scores mean that less time is needed [27]
		619.lbm_s	182	28.8		
		621.wrf_s	522	25.4		
		627.cam4_s	155	57.2		
		628.pop2_s	532	22.3		
		638.imagick_s	507	28.4		
		644.nab_s	191	91.5		
		649.fotonik3d_s	244	37.3		
		654.roms_s	245	64.2		
		600.perlbench_s	832	2.1	3.2 (INT)	
		602.gcc_s	823	4.8		
		605.mcf_s	1369	3.5		
		620.omnetpp_s	815	2.0		
		623.xalancbmk_s	444	3.2		
		625.x264_s	703	2.5		
		631.deepsjeng_s	651	2.2		
		641.leela_s	999	1.7		
		648.exchange2_s	807	3.6		
		657.xz_s	492	12.6		
PARSEC3.0	Linux/i386, Linux/AMD64, Linux/Itanium, Solaris/Sparc [33]	blackscholes	133	—	133	—
		bodytrack	346		346	
		canneal	258		258	
		facesim	771		771	
		fluidanimate	974		974	
		freqmine	776		776	
		streamcluster	2037		2037	
		swaptions	424		424	
		x264	144		144	
		dedup	58		58	
		raytrace	245		245	
		vips	179		179	
CINT2006 (speed)	Unix (AIX, HP-UX, Linux, Mac OS X, Solaris), Windows [28]	400.perlbench	742	13.2	19.6	Higher scores mean that less time is needed [27]
		401.bzip2	603	16.0		
		403.gcc	373	21.6		
		429.mcf	283	32.2		
		445.gobmk	567	18.5		
		456.hmmer	433	21.6		
		458.sjeng	880	13.7		
		462.libquantum	409	50.6		
		464.h264ref	983	22.5		
		471.omnetpp	434	14.4		
		473.astar	553	12.7		

(continued on next page)

Table 2 (*continued*).

CFP2006 (speed)	Unix (AIX, HP-UX, Linux, Mac OS X, Solaris), Windows [28]	410.bwaves	406	33.5	23.9	Higher scores mean that less time is needed [27]
		433.milc	549	16.7		
		434.zeusmp	400	22.8		
		435.gromacs	334	21.4		
		436.cactusADM	385	31.1		
		437.leslie3d	229	41.0		
		444.namd	485	16.5		
		450.soplex	278	30.0		
		453.povray	276	19.2		
		454.calculix	963	8.57		
		459.GemsFDTD	354	30.0		
		465.tonto	497	19.8		
		470.lbm	337	40.8		
		482.sphinx3	665	29.3		

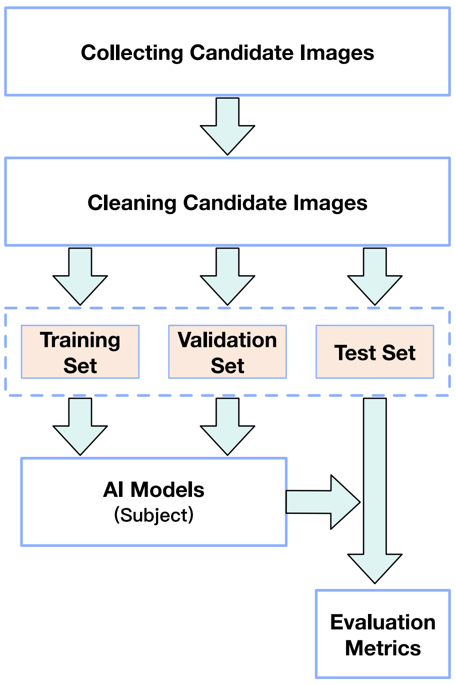


Fig. 14. The ImageNet evaluation working process.

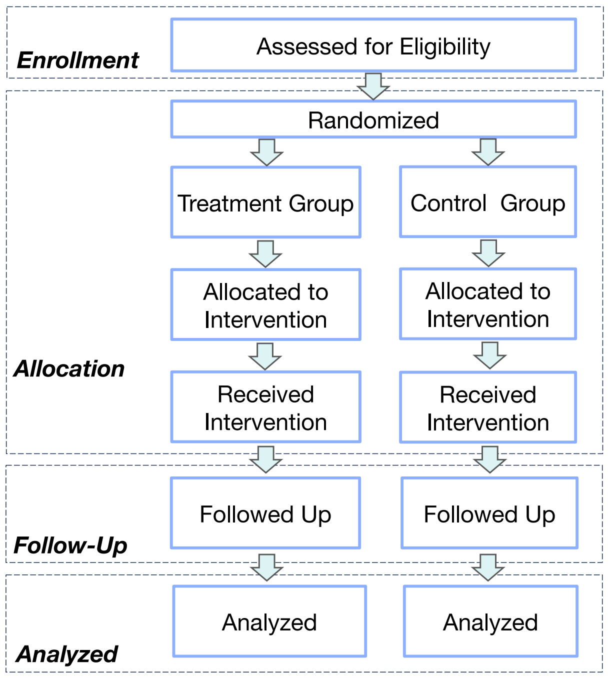


Fig. 15. The randomized controlled trials (RCT) evaluation process [40].

time and financial costs limit their application. To compensate for the shortcomings of RCTs, emerging clinical evaluation methods, such as Real-World Data (RWD) assessment and digital clinical trials, have been proposed [38,39]. These novel medical assessments are still in their early stages and have noticeable deficiencies in their theoretical foundations, such as lacking rigor and reliability.

Evaluations in the field of psychology:

In the field of psychology, social and personality psychologists often rely on scales, such as psychological inventories, tests, or questionnaires [7], to evaluate psychometric variables [7]. These variables include attitudes, traits, self-concept, self-evaluation, beliefs, abilities, motivations, goals, social perceptions, and more [7].

It is important to note that cognitive biases, which are systematic patterns of deviation from norm or rationality in judgment [41], may introduce distortions in self-report style evaluations.

5.2. *A case study on CPU benchmarks*

Within this scenario, we assess the same CPU utilizing diverse CPU benchmarks, namely SPEC CPU2006 [28], PARSEC 3.0 [33], and SPEC CPU2017 [27], which are proposed by distinct entities employing diverse methodologies.

Employing these benchmark suites, we evaluate the performance of a subject, the Intel Xeon Gold 5120T processor, and proceed to compare the resultant evaluation outcomes. During the experiments, apart from the processor itself, we provide the support system in the following manner: a 384 GB of memory, a 16TB disk, and the utilization of Ubuntu 20.04 as the operating system. To facilitate the compilation process, we employ the GNU Compiler Collection (GCC) version 9.4. We also use the largest data set of each benchmark suite (for SPEC CPU, it is a 'ref' data set, and for PARSEC, it is a 'native' data set) and run each workload three times for a comprehensive evaluation. For the SPEC CPU2006 benchmark suite, we use the default configuration file of SPECspeed Metric.

The evaluation outcomes are presented in Table 2. The discernible discrepancies observed in the evaluation outcomes can be comprehensively elucidated by taking into account the significant disparities inherent in different benchmark suites. The variations encompass the selection of distinct problem or task instances, the algorithms, the implementation of algorithms, the value functions utilized, the composite metrics employed for evaluation, the reference support system, and the reference subject, which is witnessed by Table 1.

The experimental findings illustrated in Table 2 reveal significant discrepancies in the evaluation outcomes of the same CPU when assessed using different CPU benchmark suites (the SPEC CPU and PARSEC benchmark suites). Furthermore, comparing evaluation outcomes from different evaluators becomes challenging when they employ different benchmarks. There are several reasons as follows. Firstly, they utilize different value functions. Secondly, the SPEC CPU benchmark suite encompasses a reference machine, whereas the PARSEC benchmark suite lacks such a basis for comparison. Moreover, the EECs cannot be ensured even when using the same value function and reference machine. This is because different benchmark suites introduce variations in problem or task instances, algorithms, and the implementation of algorithms.

Moreover, there are significant differences in evaluation outcomes of the same CPU, even when using the same benchmark suite with different versions. It stems from the different implementations of algorithms on varied support systems. Let us take the gcc workload in the SPEC CPU benchmark suite as an example. When evaluated under the CPU2006 benchmark suite, the CPU achieved a score of 21.6 in the 403.gcc workload, while it scored 4.8 in the 602.gcc_s workload under the SPECspeed benchmark of SPEC CPU2017. These scores show a disparity of nearly five-fold. The discrepancies in evaluation outcomes can be mainly attributed to variations in the reference machine used by the respective benchmark suites, as outlined comprehensively in Table 1. Moreover, the two workloads utilize different GCC compiler versions, with the 403.gcc workload utilizing GCC version 3.2 and the 602.gcc_s workload utilizing GCC version 4.5. Although the command flag of both workloads is ‘ref,’ the input data of the 403.gcc workload consists of nine C-code workloads, while the input data of the 602.gcc_s workload is the preprocessed GCC compiler code. Additionally, 403.gcc is not multi-threaded, while the multi-threaded is permitted for 602.gcc_s.

Furthermore, when being implemented with the same version of GCC compiler and using the same input data, the variances in evaluation outcomes for the gcc workload between the 502.gcc_r workload in SPECrate benchmark suite and 602.gcc_s in SPECspeed benchmark suite are more than twentyfold, which stem from the adoption of disparate value functions and the distinct implementations of the same algorithm. The SPECrate benchmark suite workloads are designed to assess throughput, employing multiple copies of a single-thread implementation during evaluations, while the SPECspeed benchmark suite workloads solely measure execution time, and the utilization of multiple threads is optional throughout the evaluation process. For the evaluation condition, 502.gcc_r workload makes fifty-six copies, running with a single thread, while 602.gcc_s workload has only one copy but runs with fifty-six threads.

The observed variations in evaluation outcomes for a particular CPU across different benchmark suites underscore the necessity of advancing the science and engineering of evaluation. While state-of-the-art CPU benchmarks have made significant progress, they do have certain drawbacks that need to be addressed. Firstly, the lack of comparability among evaluation results from different evaluators is a significant concern. Secondly, the significant discrepancies in evaluation outcomes cannot be traceable. Lastly, state-of-the-art CPU benchmarks often struggle to provide a realistic estimate of the parameters of real-world systems (ES) with a high level of confidence.

According to the comprehensive elements of a benchmark discussed in Section 4.1, many CPU benchmarks, such as the SPEC CPU benchmark suites, primarily focus on the reference implementation of algorithms, metrics, and references while neglecting other essential constituents and components. In the following section, we will carefully analyze and highlight the shortcomings of state-of-the-art and state-of-the-practice evaluation and benchmarks, employing our own terminology.

5.3. The reflections on state-of-the-art and state-of-the-practice benchmarks and evaluation

To further illustrate the limitations of existing evaluation and benchmarking practices, we present Fig. 16, which showcases these shortcomings within the evaluatology framework. By examining this figure, we can gain a clearer understanding of the areas where state-of-the-art and state-of-the-practice evaluation and benchmarks fall short.

It is evident that *a lack of consensus exists regarding concepts and terminologies across different areas of study*. This lack of consensus often leads to confusion and misinterpretation, especially when the same terms are used in different disciplines with varying meanings.

For example, the term “benchmark” is commonly employed in computer science, finance, and business disciplines but without a formal definition. Moreover, even within these fields, the definition of “benchmark” can be vague and subject to interpretation. In contrast, psychology may use the term “scale” as a concept similar to benchmark, while social science and medicine may not have an analogous concept at all.

Recognizing this challenge, our work has aimed to propose universal concepts and terminologies that can bridge these disciplinary gaps. By establishing clear and standardized definitions, we seek to promote a shared understanding and facilitate effective communication and collaboration across different areas of study.

Few works discuss the essence of evaluation, let alone reaching a consensus on it. Evaluation is often mistakenly equated with measurement or testing without clear differentiation. For instance, in computer science and psychology, evaluation and measurement are often used interchangeably. In the context of testing, where the goal is to determine whether an individual or a system aligns with the expected behavior defined by test oracles, evaluation is often conflated with testing. For instance, according to the SPEC terminology, a benchmark refers to “a test, or set of tests, designed to compare the performance of one computer system against the performance of others” [42,43]. SPEC is a highly influential benchmark organization. Our work has revealed the essence of the evaluation.

The proposed evaluation theories and methodologies are often domain-specific, with a lack of universally applicable foundational principles and evaluation methodologies that transcend diverse disciplines. Different disciplines do not delve into the underlying principles of evaluation. Instead, they adopt a pragmatic approach and prioritize guidelines for conducting evaluations within specific contexts.

For instance, in the medical discipline, the focus is primarily on eliminating confounding variables within the specific groups or cohorts being studied. In the business discipline, efforts are concentrated on searching the state of the practice.

The most rigorous theoretical foundation can be found in the field of clinical trials. For instance, Randomized Controlled Trial (RCT) techniques are employed to rule out the effect of confounding variables. However, there is a lack of universal problem formulations or fundamental solutions that fully consider the intricate interactions among the key components of EMs in diverse scenarios.

There are two serious drawbacks to the RCT methodology and its variants. Firstly, there is a lack of a stringent hierarchical definition of EC and EECs. The variations in ECs can introduce confounding that may affect the results and make meaningful comparisons difficult. Without ensuring EECs, it becomes an illusion to expect comparable evaluation outcomes.

Secondly, when it comes to studying complex systems such as human beings or experimental animals, which we refer to as support systems, the RCT methodology and its variants may struggle to establish an REM. This kind of support system is characterized by a multitude of independent variables, making it difficult to isolate and control all relevant factors in a controlled experimental setting. Consequently, it becomes challenging to eliminate confounding variables and ensure unbiased evaluation outcomes completely.

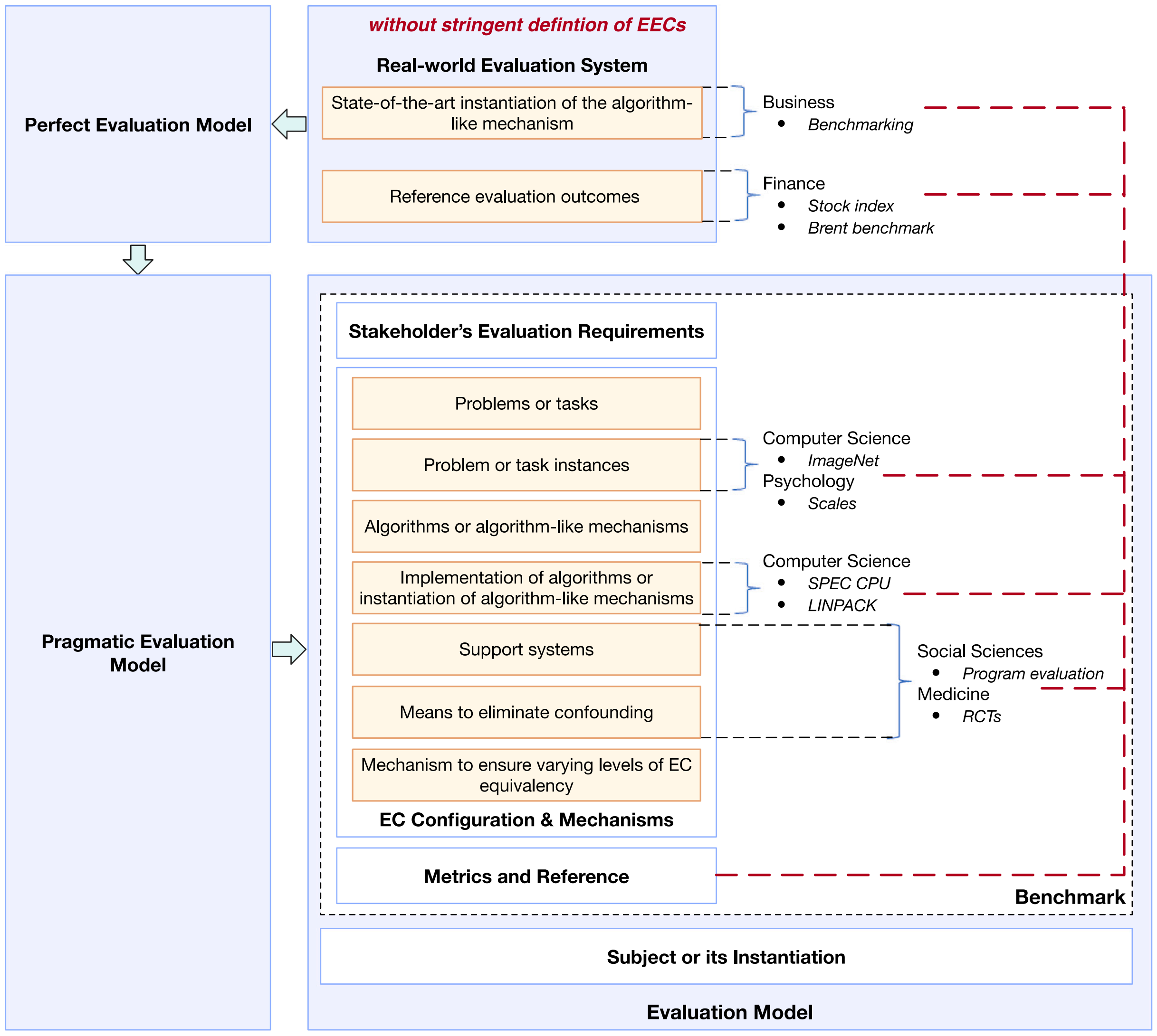

without stringent defintion of EECs
Real-world Evaluation System
State-of-the-art instantiation of the algorithm-like mechanism
Reference evaluation outcomes
Perfect Evaluation Model
Business
Benchmarking
Finance
Stock index
Brent benchmark
Pragmatic Evaluation Model
Stakeholder's Evaluation Requirements
Problems or tasks
Problem or task instances
Algorithms or algorithm-like mechanisms
Implementation of algorithms or instantiation of algorithm-like mechanisms
Support systems
Means to eliminate confounding
Mechanism to ensure varying levels of EC equivalency
EC Configuration & Mechanisms
Metrics and Reference
Benchmark
Computer Science
ImageNet
Psychology
Scales
Computer Science
SPEC CPU
LINPACK
Social Sciences
Program evaluation
Medicine
RCTs
Subject or its Instantiation
Evaluation Model

Fig. 16. The reflections on state-of-the-art and state-of-the-practice benchmarks and evaluation are based on the science and engineering of evaluation.

In the realms of business and finance, different observational study methodologies are widely used, and we have revealed their inherent limitations in Section 2.1. An observational study is not even an experiment. Certainly, it cannot eliminate confounding variables and reveal the cause-and-effect relationships. In the business discipline, benchmarking assumes the state-of-the-art instantiation of the algorithm-like mechanism and the reference evaluation outcomes. In finance and education disciplines, benchmarks or indexes assume the role of reference evaluation outcomes in an observational study that measures variables of interest but does not attempt to influence the response [2].

Rossi et al. [1] propose a valuable framework for evaluating methodologies in the field of social science. However, they do not provide a universal theory that can be applied to different disciplines. Their limitations stem from their narrow focus on assessing social programs without developing a generalized theory for evaluating other subjects in complex conditions.

Rossi et al. indeed utilized or developed some approaches to isolate the social programs' impacts, e.g., comparison group designs and randomized controlled trials (RCT), but they failed to explicitly state the underlying principles and methodology for universal science and engineering of evaluation.

Within the computer science field, there are varying viewpoints and perspectives. For example, Hennessy et al. [4] highlight the significance of benchmarks and define them as programs specifically selected for measuring computer performance. On the other hand, John et al. [3] compile a book on performance evaluation and benchmarking without providing formal definitions for these concepts. Kounev et al. [42] present a formal definition of benchmarks as "tools coupled with methodologies for evaluating and comparing systems or components based on specific characteristics such as performance, reliability, or security". The ACM SIGMETRICS group [5,6] considers performance evaluation as the generation of data that displays the frequency and execution times of computer system components, with a preceding orderly and well-defined set of analysis and definition steps.

In psychology, social and personality psychologists often utilize scales, such as psychological inventories, tests, or questionnaires, to assess psychometric variables [7,7]. While these tools are commonly used, it is important to recognize that they rely on virtual assessments and self-report-style evaluations, which may introduce potential distortions.

To overcome this limitation, we suggest implementing a physical application of an EC to the subjects, supplemented with a variety of measurement instruments. This approach aims to provide a more objective and accurate assessment of various aspects, including attitudes, traits, self-concept, self-evaluation, beliefs, abilities, motivations, goals, and social perceptions [7], by incorporating tangible and observable data.

Various disciplines have proposed engineering approaches to evaluations. However, they fail to provide universal benchmark concepts, theories, principles, and methodologies.

For instance, benchmarks are commonly utilized in finance, computer science, and business, albeit with inconsistent meanings and practices. Regrettably, there have been limited discussions in previous works regarding universal benchmark principles and methodologies that can be applied across different disciplines. From a computer science standpoint, Kounev et al. [42] provide a comprehensive foundation for benchmarking, including metrics, statistical techniques, experimental design, and more.

Most state-of-the-art and state-of-the-practice benchmarks overlook an essential aspect: the stakeholders' evaluation requirements. This oversight leads to a failure to consider different and diverse evaluation requirements. For instance, they do not enforce the discrepancy threshold in evaluation outcomes, nor do they consider evaluation confidence, among other crucial factors. As a result, most CPU benchmarks are ill-equipped to meet the evaluation requirements in scenarios involving safety-critical, mission-critical, and business-critical applications.

Another issue is the lack of a stringent definition for similar concepts, such as an EECs or LEECs. For example, most CPU or AI (deep learning) benchmarks, like ImageNet, fail to provide a clear definition of an EECs or LEECs. Instead, they jump directly into the implementation of algorithms or a specific dataset labeled with the ground truth without proper justification. Additionally, the support system, which plays a crucial role in some cases, is omitted without any explanation of the condition of simplifying the benchmarks. Furthermore, most of the methodologies fail to discuss the confounding elimination mechanism. This oversight can potentially introduce bias and inaccuracies in the evaluation outcomes.

Not surprisingly, the intricate evaluation mechanisms and policies introduced in Section 4.4.2 are not explicitly discussed in the design and implementation of most benchmarks. For instance, it fails to address important aspects such as investigating and characterizing real-world ES, the design and implementation of a perfect EM, the modeling policy and procedure from a real-world ES to an EM, and the sampling policy and procedure from a perfect EC to a pragmatic EC. This omission makes it difficult for the benchmark to adapt to intricate evaluation scenarios.

It is crucial to include these mechanisms and policies to ensure the benchmark's applicability and effectiveness in complex evaluation scenarios. Without explicit discussion of the real-world ES, it is difficult to establish an EC that captures the characteristics and requirements of real-world evaluations. Furthermore, exploring different sampling and modeling policies is essential to gain the confidence of the evaluation community in using the benchmark for inferring parameters of real-world ES. By carefully designing these policies, we can strike a balance between achieving high accuracy in evaluation outcomes and managing the associated evaluation costs.

There are many widely used AI (deep learning) benchmarks. Taking the ImageNet dataset as an illustrative example [31], we reveal their limitations. Firstly, a specific AI benchmark like ImageNet cannot be traced back to an explicit formulation of a problem or task and instead manifests itself in the form of a dataset containing ground truth, which may possess certain biases. In other scenarios, we also encounter challenges in identifying a precise mathematical function that accurately models the chemical and biological activities within the human body (Case Three in Section 3.5.2) or the social dynamics within the target population (Case Four in Section 3.5.2). Secondly, the benchmark relies on an unverified assumption that the data distribution within the real world closely aligns with that of the collected dataset to a considerable extent. Thirdly, in real-world applications, we use the statistic of a sample – a specific benchmark – to infer the parameters of the entire population. However, we do not know their confidence levels and intervals.

5.4. What is the benefit of evaluatology?

Evaluatology serves as the foundational theory that encompasses evaluations in various fields of study. It provides a universal framework for optimizing the evaluation process, with four fundamental axioms serving as its basis. Formulating the core evaluation issues mathematically presents opportunities for seeking optimal solutions based on theoretical grounds. As a subdivision of evaluatology, benchmarkology offers a comprehensive engineering approach and methodology for evaluation, applicable across diverse disciplines.

Together, evaluatology and benchmarkology contribute reusable knowledge to different domains, encompassing universal terminology, principles, and methodologies. By sharing this knowledge base, they facilitate advancements in both the state-of-the-art and state-of-the-practice of evaluation in various realms. This unification of communities embarks on a collective journey to address future challenges.

5.5. The differences between evaluation, measurement and testing

Drawing on the preceding analysis, this subsection elucidates the marked disparity between evaluation, measurement, and testing.

First and foremost, it is important to acknowledge that measurement or testing serves as a preliminary constituent within the broader framework of evaluation. In addition to measurement and testing, an evaluation encompasses a series of steps, which we have discussed in Section 3. These steps involve defining and applying evaluation conditions to a diverse range of subjects, which ultimately leads to the creation of an evaluation model or system. Once the evaluation model or system is established, the impacts of different subjects can be inferred through the process of measuring and/or testing.

Furthermore, it is crucial to recognize that the measurement results are of an objective nature, assuming the existence of an inherent truth value for each measured quantity. Similarly, testing results also possess an objective nature as they typically yield either a positive or negative outcome for each test conducted.

Conversely, evaluation results possess a certain degree of subjectivity, such as the formulation of value functions based on the underlying measurement data, which we have discussed in the first evaluation axiom in Section 3.3.

By virtue of the aforementioned reasons, we can assert that metrology or testing serves as but one foundational aspect in the realm of evaluations.

6. Conclusion

This article formally introduces evaluatology, a discipline encompassing both the science and engineering of Evaluations. Our contributions are three-fold.

First, in order to promote consistency and facilitate cross-disciplinary understanding, we propose the adoption of universal evaluation concepts and terminologies centered around evaluation conditions.

Secondly, we reveal the essence of evaluation and propose five evaluation axioms as the foundational evaluation theory. Furthermore, we introduce the universal evaluation theory, principles, and methodology that govern the field of evaluation.

We create evaluation conditions with different levels of equivalency and apply them to diverse subjects to establish reference evaluation models that alter a single independent variable at a time while keeping all other variables as controls. We discover that the key to effective and efficient evaluations in various complex scenarios lies in establishing a series of evaluation models that maintain transitivity.

Third, building upon the science of engineering, we formally define a benchmark as a simplified and sampled evaluation condition that ensures different equivalency levels. We present a benchmark-based universal engineering of evaluation across different disciplines, which we refer to as benchmarkology.

CRediT authorship contribution statement

Jianfeng Zhan: The science and engineering of evaluation, encompassing the universal evaluation concepts, terminologies, theory and methodology, the universal benchmark concepts, principles, theories, and methodologies, and mathematical formulation of fundamental evaluation issues. **Lei Wang:** Whole-session discussion, the axiom of the true evaluation results, the CPU benchmark experiments in Section 5.2, the summary of the related work for finance and HPC benchmarks in Section 5.1, and the mathematical formulation of fundamental evaluation issues. **Wanling Gao:** Presentations of all figures, whole-session discussion, and the discussion of the benchmark inspired the first author to think about what is essential of the benchmark. **Hongxiao Li:** Whole-session discussion, mathematics notations, mathematical formulations of three fundamental issues in Evaluatology, and mathematical formulations of EECs and LEECs. **Chenxi Wang:** Whole-session discussion, design, implementation, analysis, and presentation of experiments in Section 5.2. **Yunyou Huang:** Discussion, and the summary of the related work of drug evaluations in Section 5.1. **Yatao Li:** Discussion, and the mathematical formulations of evaluation traceability in Section 3.7.3. **Zhengxin Yang:** Discussion and presentation of metrology, and real-world evaluation of safety-critical AI systems. **Guoxin Kang:** Discussions, and related work about business benchmarking in Section 5.1. **Chunjie Luo:** Discussions. **Hainan Ye:** Discussions. **Shaopeng Dai:** Presentations of some figures. **Zhifei Zhang:** Discussions, and the summary of related work of drug evaluations.

Declaration of competing 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] P.H. Rossi, M.W. Lipsey, G.T. Henry, Evaluation: A Systematic Approach, Sage publications, 2018.
[2] D.S. Starnes, D. Yates, D.S. Moore, The Practice of Statistics, Macmillan, 2010.
[3] L.K. John, L. Eeckhout, Performance Evaluation and Benchmarking, CRC Press, 2018.
[4] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, Elsevier, 2011.
[5] J.C. Browne, An analysis of measurement procedures for computer systems, ACM SIGMETRICS Perform. Eval. Rev. 4 (1975) 29–32.
[6] M.E. Knudson, A performance measurement and system evaluation project plan proposal, ACM SIGMETRICS Perform. Eval. Rev. 13 (1985) 20–31.
[7] M. Furr, Scale construction and psychometrics for social and personality psychology, Scale Constr. Psychometr. Soc. Pers. Psychol. (2011) 1–160.
[8] R.C. Camp, Benchmarking: The Search for Industry Best Practices that Lead to Superior Performance, Asq Press, 1989.
[9] L. Fisher, Some new stock-market indexes, J. Bus. 39 (1966) 191–225.
[10] H.D. Young, R.A. Freedman, L.A. Ford, University physics with modern physics, 2020.
[11] J. Stewart, Single Variable Calculus: Concepts and Contexts, Cengage Learning, 2018.
[12] System, 2024, https://www.merriam-webster.com/dictionary/system. (Accessed 6 February 2024).
[13] A. Backlund, The definition of system, Kybernetes 29 (2000) 444–451.
[14] I. BiPM, I. IFCC, I. IUPAC, O. ISO, The international vocabulary of metrology—basic and general concepts and associated terms (vim), Joint Comm. Guides Metrol. 200 (2012) 2012.
[15] R.N. Kacker, On quantity, value, unit, and other terms in the jcgm international vocabulary of metrology, Meas. Sci. Technol. 32 (2021) 125015.
[16] S.S. Stevens, On the theory of scales of measurement, Science 103 (1946) 677–680.
[17] L. Baresi, M. Young, Test oracles, 2001.
[18] D. Choudhary, V. Kumar, Software testing, J. Comput. Simul. Model. 1 (2011) 1.
[19] J.A. Whittaker, What is software testing? And why is it so hard? IEEE Software 17 (2000) 70–79.
[20] P.J. Denning, The science of computing: What is computer science? Am. Sci. 73 (1985) 16–19.
[21] D.S. Johnson, A catalog of complexity classes, in: Algorithms and Complexity, Elsevier, 1990, pp. 67–161.
[22] S.B. Cooper, Computability Theory, Chapman and Hall/CRC, 2017.
[23] R. Sedgewick, P. Flajolet, An Introduction to the Analysis of Algorithms, Addison-Wesley Longman Publishing Co. Inc, 1996.
[24] K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, et al., The landscape of parallel computing research: A view from Berkeley, 2006.
[25] B. Andersen, T. Fagerhaug, S. Randmæl, J. Schuldmaier, J. Prenninger, Benchmarking supply chain management: finding best practices, J. Bus. Ind. Mark. 14 (1999) 378–389.
[26] D.P. Scheitrum, C.A. Carter, C. Revoredo-Giha, Wti and brent futures pricing structure, Energy Econ. 72 (2018) 462–469.
[27] SPEC, SPEC CPU2017 benchmark suite, 2017, https://www.spec.org/cpu2017/.
[28] SPEC, SPEC CPU2006 benchmark suite, 2006, https://www.spec.org/cpu2006.
[29] SPEC, SPEC CPU Benchmark Suite. https://www.spec.org/benchmarks.html#cpu.
[30] J.J. Dongarra, P. Luszczek, A. Petitet, The linpack benchmark: past, present and future, Concurr. Comput. Pract. Exp. 15 (2010) 803–820.
[31] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 248–255.
[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.
[33] C. Bienia, S. Kumar, J.P. Singh, K. Li, The parsec benchmark suite: Characterization and architectural implications, in: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, 2008, pp. 72–81.
[34] D.F. Cavers, The food, drug, and cosmetic act of 1938: its legislative history and its substantive provisions, Law Contemp. Probs. 6 (1939) 2.
[35] J. Jenkins, S. Hubbard, History of clinical trials, in: Sem. Oncol. Nurs., 1991, pp. 228–234.
[36] J.W. Sellers, C.M. Mihaescu, K. Ayalew, P.D. Kronstein, B. Yu, Y.M. Ning, M. Rodriguez, L. Williams, N.A. Khin, Descriptive analysis of good clinical practice inspection findings from us food and drug administration and european medicines agency, Ther. Innov. Regul. Sci. 56 (2022) 753–764.
[37] S. Monti, V. Grosso, M. Todoerti, R. Caporali, Randomized controlled trials and real-world data: differences and similarities to untangle literature data, Rheumatology 57 (2018) vii54–vii58.
[38] O. Inan, P. Tenaerts, S. Prindiville, H. Reynolds, D. Dizon, K. Cooper-Arnold, M. Turakhia, M. Pletcher, K. Preston, H. Krumholz, et al., Digitizing clinical trials, NPJ Digit. Med. 3 (2020) 101.
[39] S.V. Ramagopalan, A. Simpson, C. Sammon, Can real-world data really replace randomised clinical trials? BMC Med. 18 (2020) 1–2.
[40] D. Moher, K.F. Schulz, D.G. Altman, C. Group*, The CONSORT statement: revised recommendations for improving the quality of reports of parallel-group randomized trials, Ann. Intern. Med. 134 (2001) 657–662.
[41] BIAS, FOC, The evolution of cognitive bias, Handb. Evol. Psychol. (2015) 2.
[42] S. Kounev, K.D. Lange, J.Von. Kistowski, Systems Benchmarking, Springer, 2020.
[43] SPEC, SPEC glossary, 2023, https://www.spec.org/spec/glossary.

Dr. Jianfeng Zhan is a Full Professor at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and University of Chinese Academy of Sciences (UCAS). He is also the director of the Research Center for Distributed Systems at ICT, CAS. He holds a B.E. in Civil Engineering and an MSc in Solid Mechanics from Southwest Jiaotong University, obtained in 1996 and 1999 respectively. He completed his Ph.D. in Computer Science from the Institute of Software, CAS, and UCAS in 2002.
Dr. Zhan's research encompasses a wide range of areas, including chips, systems, and benchmarks. His work involves benchmarking, designing, implementing, and optimizing diverse systems. He has successfully translated his academic research into advanced technologies that have had a significant impact on general-purpose production systems. His team's innovations and research results, including 35 patents, have been adopted in benchmarks, operating systems, and cluster and

cloud system software, contributing to the advancement of parallel and distributed systems in China and globally.

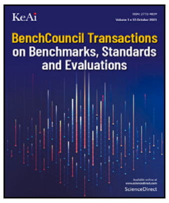
Dr. Zhan is the founder and chair of the International Open Benchmark Council (BenchCouncil), a prominent organization in the field of benchmark and Evaluation. He also serves as the Co-Editor-in-Chief of TBench alongside Prof. Tony Hey. From 2018 to 2022, Dr. Jianfeng Zhan served as an Associate Editor for the IEEE TPDS journal. He has received several accolades for his work, including the second-class Chinese National Technology Promotion Prize in 2006, the Distinguished Achievement Award of the Chinese Academy of Sciences in 2005, and the IISWC Best Paper Award in 2013.

Throughout his career, Dr. Zhan has supervised numerous graduate students, post-doctoral researchers, and engineers, with a total of over ninety individuals under his guidance.

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/



Full length article

An approach to workload generation for modern data centers: A view from Alibaba trace

Yi Liang [a,*], Nianyi Ruan [a], Lan Yi [b], Xing Su [a]

[a] *Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China*
[b] *Yunhe Enmo (Beijing) Information Center, Beijing 100080, China*

ARTICLE INFO

Keywords:
Workload generation
Data center
Trace

ABSTRACT

Modern data centers provide the foundational infrastructure of cloud computing. Workload generation, which involves simulating or constructing tasks and transactions to replicate the actual resource usage patterns of real-world systems or applications, plays essential role for efficient resource management in these centers. Data center traces, rich in information about workload execution and resource utilization, are thus ideal data for workload generation. Traditional traces provide detailed temporal resource usage data to enable fine-grained workload generation. However, modern data centers tend to favor tracing statistical metrics to reduce overhead. Therefore the accurate reconstruction of temporal resource consumption without detailed, temporized trace information become a major challenge for trace-based workload generation. To address this challenge, we propose STWGEN, a novel method that leverages statistical trace data for workload generation. STWGEN is specifically designed to generate the batch task workloads based on Alibaba trace. STWGEN contains two key components: a suite of C program-based flexible workload building blocks and a heuristic strategy to assemble building blocks for workload generation. Both components are carefully designed to reproduce synthetic batch tasks that closely replicate the observed resource usage patterns in a representative data center. Experimental results demonstrate that STWGEN outperforms state-of-the-art workload generation methods as it emulates workload-level and machine-level resource usage in much higher accuracy.

1. Introduction

With the rapid evolution of computing and networking technologies, modern data centers have become the primary infrastructural of cloud computing [1–3]. A fundamental challenge in modern data centers is how to efficiently utilize the resources through resource management [4–7]. This encompasses various techniques, including capacity planning, resource scheduling, and resource isolation [8–13]. The essence of resource management is to schedule workloads and allocate resources for maximized resource utilization. In order to refine the strategy of resource allocation, we should always understand the resource consumption of workload first. However, the diversity of workloads and the restricted access to underlining source code prevent us from in-depth analyzing of the hosted applications. As a result, the generation of synthetic workloads which mimic the resource consumption of cloud applications has become a critical research area [14–17]. The ultimate goal of workload generation is to replicate the resource consumption patterns observed in real data center applications.

In cloud data centers, tracing systems can meticulously capture the resource consumption and execution dynamics of workloads, subsequently producing the trace data [18–20]. Such trace collection has led to the prevailing role of trace-based workload emulations in the domain [16,21]. The trace data thoroughly record resource utilization of applications at each monitoring cycle, providing detailed temporal insights to accurately emulate workloads. The trace-based workload generation first extracts resource consumption patterns from trace data, then constructs modular building blocks, and finally assembles building blocks to emulate the original resource consumption patterns.

Previous research on workload generation are mainly conducted in the coarse-grained manner [22–26]. They use some pre-defined benchmark applications (e.g., TPC bench [27]) as building blocks to generate the synthetic workloads that have the similar resource usage statistics (such as the average, maximum statistics and the probability distributions) as those recorded in trace data. Such approaches fall short in faithfully replicate the workload's resource usage sequence along execution, hence are not good at accurately reproducing the temporal resource utilization patterns in data centers. This limitation hampers their capability in supporting fine-grained resource scheduling in data centers.

* Corresponding author.
E-mail address: yliang@bjut.edu.cn (Y. Liang).

https://doi.org/10.1016/j.tbench.2024.100164
Received 17 March 2024; Received in revised form 2 May 2024; Accepted 11 May 2024
Available online 21 May 2024

Fine-grained workload generation has emerged as a prominent solution [28,29]. These methods strive to precisely replicate the intricate details of a workload's resource utilization throughout every moment of its runtime. They heavily depend on the trace data, which captures the workload's explicit temporal resource usage patterns and micro-architectural behaviors [18], to mimic workloads that exactly correspond to their recorded resource usage sequences in the trace. However, as the scales of data centers continue to expand, the complexity of trace information rises exponentially, which often results in substantial resource and time cost during the trace collection [30]. To minimize the overhead, modern data centers tend to trace statistical metrics rather than detailed temporal consumption and granular workload behaviors. For instance, in the Alibaba clusterdata2018 trace (Alibaba trace, in short) [19], the most recent large-scale data center trace, the resource usage of batch workloads (namely, batch tasks) is merely recorded as the maximum and average statistics. Consequently, the challenge of accurately reconstructing temporal resource consumption of workloads without the detailed and temporalized trace information remains an open issue.

To address the challenge, we propose Statistical Trace-based Workload Generation (STWGEN) as the innovative method to leverage statistical trace data for workload generation. STWGEN is originally designed to generate batch task workloads based on Alibaba trace, it can also easily be extended for more general workload generation based on statistical trace data. STWGEN is comprised of two components: a suite of C program-based [31] flexible workload building blocks and a heuristic strategy to assemble building blocks for workload generation. STWGEN integrates the two components with a sophisticated workload submission mechanism, enabling the generation of synthetic workloads that can faithfully reproduce the observed resource consumption patterns in modern data centers. Our main contributions are as follows:

Firstly, with in-depth analysis of Alibaba trace, we characterized the resource usage patterns of batch task workloads and identified some typical and helpful features, including the weak correlation between CPU and Memory usage, the significant variation in CPU usage, the stable memory consumption and the differentiated resource usage reproduction demands among tasks.

Secondly, we developed fundamental workload building blocks to simulate CPU and memory usage respectively. These blocks are designed as C programs to do parameterized amount of computation and memory allocation operations for replicating CPU and memory usage of given scale. Moreover, the building blocks can dynamically adjust resource utilization in execution.

Thirdly, we propose a heuristic strategy to reconstruct the temporal resource usage of batch tasks. The strategy integrates the Simulated Annealing algorithm and the JAYA algorithm to find the optimal solution for reconstructing the maximum and average resource utilization statistics of batch tasks and reproducing the machine-level resource utilization.

Lastly, we performed a thorough evaluation on STWGEN. The experimental results demonstrate that, based on Alibaba trace, STWGEN can generate batch task workloads with a average deviation of less than 14.1% on average and maximum task-level resource usage statistics, and a average deviation of less than 14.3% on machine-level total resource usage. Compared to the state-of-art methods, STWGEN achieves up to 98.6% reduction in the above deviations.

The remaining sections of the paper are organized as follows: Section 2 introduces the Alibaba Trace and formulates the workload generation problem. Section 3 characterizes the workloads in Alibaba Trace. Section 4 elaborates on the proposed STWGEN method. Section 5 is dedicated to the evaluation of STWGEN method. Section 6 reviews the literature and Section 7 concludes the paper.

2. Background and problem formulation

We take Alibaba trace as the target for the workload generation. We first describe the detail information of Alibaba trace and formally define the trace-based workload generation problem.

2.1. Alibaba trace

Released in 2018, the Alibaba trace [18,32,33] captures the workload behavior and resource usage of a production cluster within Alibaba, comprising approximately 4,000 physical servers, over a period of eight days. Both online services and batch jobs are co-located in this cluster. We focus on the batch workload generation in this paper. In Alibaba cluster, one batch job consists of one or more batch tasks. A batch task can be executed in parallel on multiple machines, known as task instances in Alibaba trace. Batch tasks are the basic execution units of a batch workload and also the resource consumption elements. Reproducing task-level resource utilization is the cornerstone for conducting a fine-grained analysis of cluster resource usage patterns.

In Alibaba trace, the batch task, online service (hosted in containers) and *machine_usage* resource usage data are recorded in *batch_instance*, *container_usage* and machine-usage table, respectively. Among these tables, machine-usage and *container_usage* capture the respective resource usage information every 30 s with the aligned timestamps. However, *batch_instance* only logs the average and maximal statistics of batch tasks' resource usage during their executions. In addition to the resource usage statistics for each individual batch task, the total amount of resource usage of concurrent tasks executed on a machine at a recording time point can be derived by substracting the container resource usage data in *container_usage* table from the server resource usage data in *machine_usage* table at that specific moment. These two types of information constitute the basis for generating the batch task workload.

2.2. Problem definition

Our work focuses on the batch task workload generation based on Alibaba trace. We aim to reproduce the usage of two primary resources that batch workload consume: CPU and memory. The problem is formulated as follows.

Given a data center with massive machines, the resource usage of an individual machine at time point i, can be described as $un_i = (uc_i, um_i)$, where, uc_i and um_i represent its CPU and memory usage at time i, respectively. During time period T, there are N batch tasks executed on this machine, denoting as $TS = (ts_1, ts_2, \ldots, ts_N)$. Each task can be represent as $ts_j = (cavg_j, mavg_j, cmax_j, mmax_j, st_j, et_j)$, where, $cavg_j$ and $mavg_j$ are the average CPU and memory usage of task j, respectively.$cmax_j$ and $mmax_j$ are the maximal CPU and memory usage of task j, respectively. st_j and et_j are the starting time and ending time of task j, respectively. According to the starting and ending time of tasks, at any specific recording time point i, there is a subset of tasks executed concurrently on a machine, denoted as $CT_i, CT_i \sqsubseteq \mathbf{TS}$.

Workload generation in this paper is to construct the synthetic workloads for tasks in $\mathbf{TS}$, each with a dynamic resource usage sequence during execution, denoted as $rs_i = r_{i,st_i}, \ldots, r_{i,et_i}$, where, $r_{i,j} = (rc_{i,j}, rm_{i,j})$. represent the CPU and memory usage of task i at the jth time point. We define $DIV_{task}(\cdot)$ as the deviation of the resource usage statistics of the generated batch task workload from the corresponding statistical data recorded in the trace, and $DIV_{machine}(\cdot)$ as the deviation of the cumulative resource usage of all concurrently executed synthetic batch task workloads on a machine from their corresponding machine-level total resource usage recorded in the trace. Our goal can be expressed as follows:

$$min(DIV_{task}(ts_j, rs_j)), j \in [1, N] \tag{1}$$

$$min(DIV_{machine}(un_i, CT_i)), i \in T \tag{2}$$

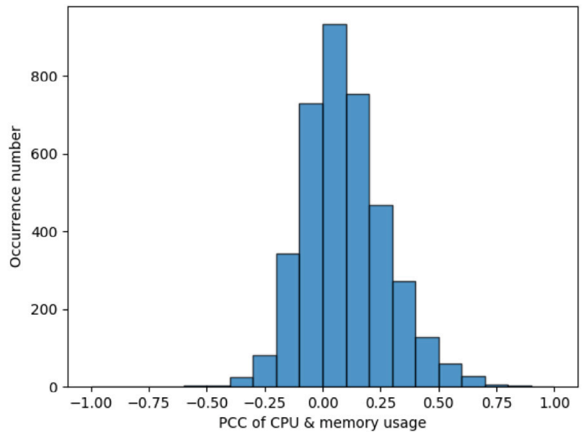


Fig. 1. Correlation between CPU and memory Usage.

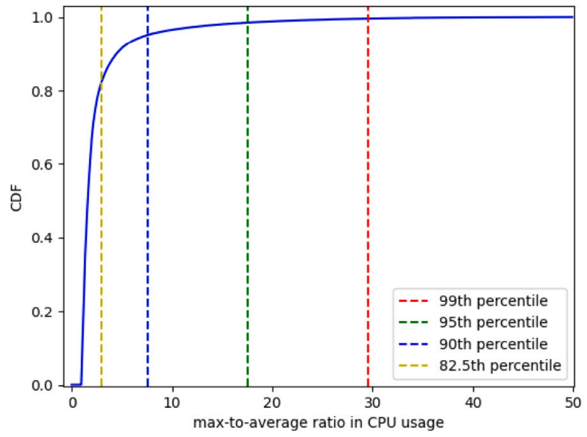


Fig. 2. Distribution of max-to-average ratio on CPU usage.

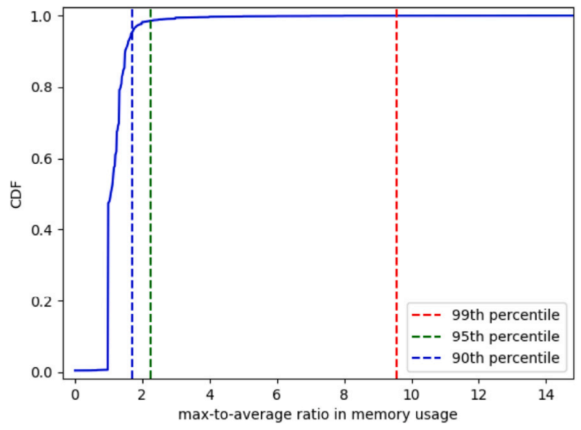


Fig. 3. Distribution of max-to-average ratio on memory usage.

3. Insight from Alibaba trace

In this section, we first conduct quantitative observations on the resource usage of batch tasks in Alibaba trace, and then extract insights into workload generation from these observations. Our observations are based on the *batch_instance* table in Alibaba trace.

Observation #1: Relatively Weak Correlation between CPU and Memory Usage

To analyze whether the CPU and memory usage during batch task executions exhibit a strong correlation, we adopt Pearson Correlation Coefficient (PCC) measurement [34]. Specifically, we randomly partition all batch tasks recorded in Alibaba trace into groups, each with around 50,000 tasks. Within each group, we measure the PCC between the batch tasks' average CPU usage and average memory usage. The statistical result is shown in Fig. 1. The average absolute PCC value across all groups is 0.09, and more than 97.2% groups have the absolute PCC value less than 0.4 (which is the typical threshold for the moderate correlation). The result statistically proves a weak correlation between CPU usage and memory usage during the batch task execution.

Guide for design: This observation inspires us to employ CPU-intensive and memory-intensive operations to independently simulate the CPU and memory usage of batch tasks. By accurately simulating the usage of each resource, we can generate a complete synthetic task workload by assembling the employed operations and making some slight refinements.

Observation #2: Significant Variation in the CPU usage

With only the statistical data available, we adopt the max-to-average ratio to quantify the batch task's CPU usage variation during runtime. Intuitively, a higher ratio represents a greater variation. Fig. 2 demonstrates the CDF (Cumulative Distribution Function) of the max-to-average ratio of all batch tasks in Alibaba trace, with the 82.5th percentile being 3, the 90th percentile being 7.56, the 95th percentile being 17.48 and the 99th percentile being 29.55. This result indicates that a considerable portion of tasks experience significant fluctuations in CPU resource usage during their runtime, and some tasks undergo a surge in CPU resource utilization.

Guide for design: The significant variation of CPU usage necessitates that the generated task workload be able to dynamically and agilely produce the varying CPU resource consumptions during runtime. Further, the great gap between the maximum and average statistics points to a large search space when reconstructing the task's resource usage sequence. A computationally-efficient algorithm is thus required to generate task workloads that conform to both workload and machine-level resource usage patterns recorded in the trace.

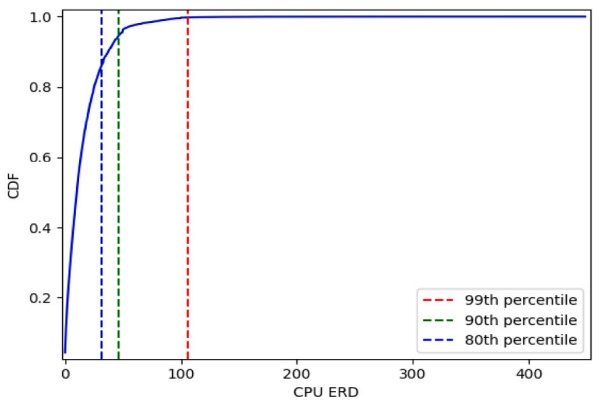
Observation #3: Low and stable Memory Usage

In the Alibaba trace, the batch task's average memory usage value ranges from 0 to 100, representing the percentage of total memory utilized on its host machine. We analyzed its distribution and found that the 90th, 95th, and 99th percentile values are 0.35, 0.77 and 2.85, respectively. Furthermore, we collected data on the max-to-average ratio of memory usage. As depicted in Fig. 3, less than 10% of batch tasks exhibit a ratio exceeding 1.69, and the 95th percentile value stands at 2.24. These findings indicate that, in comparison to CPU usage, the batch task's memory usage is relatively stable. Even though some tasks have a max-to-average ratio greater than 9.56 (that is, at the 99th percentile), their absolute variations in memory consumption remains small due to the low average base.

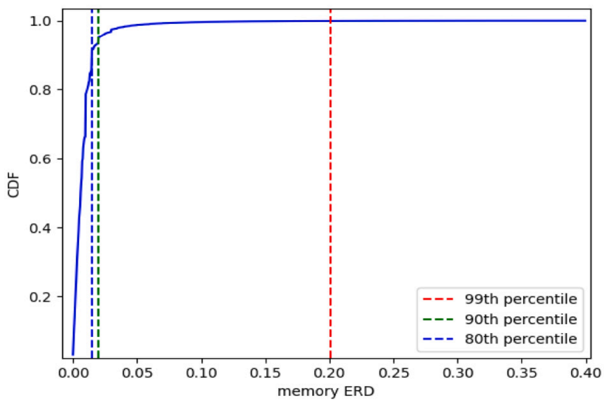
Guide for design: Due to the low memory usage of batch tasks, we should carefully select memory-intensive synthetic workload operations with minimal resource overhead to prevent simulation deviations. In addition, given the memory usage stability, the selected operations should maintain consistent memory occupancy so as to minimize the overhead associated with frequent memory allocation requests.

Observation #4: Differentiated Early Reproduction Demands among tasks

To accurately replicate the resource usage pattern of the batch task, it is crucial to explicitly reproduce the maximum statistic at least once during the task's execution. Specifically, in the context of workload generation for concurrent tasks running on a particular machine, tasks that exhibit high resource usage peaks and have short execution times should strive to reproduce their peak resource usage promptly during the early stages of their execution, so as to mitigate the risk of such tasks being unable to reach their peak usage later on due to the time point-wise machine-level total resource usage constraints recorded in the trace. To analyze the demand for early peak resource



(a) CPU



(b) memory

Fig. 4. Distribution of ERD.

usage reproduction among batch tasks, we utilize a metric called Early Reproducing Demand (ERD). ERD is defined as the ratio of the batch task's maximal resource usage to its execution duration. The formula of ERD is as follows:

$$ERD = \frac{max\_usage}{exec\_duration} \tag{3}$$

Where, max_usage is the batch task's maximal resource usage, $exec\_time$ is its execution duration (measured in second). Fig. 4 depicts the distributions of the Early Reproducing Demand (ERD) for CPU and memory resources. It clearly shows that 80% of the tasks have an ERD for CPU resources falling below a threshold 31.33, suggesting a lower need for prompt peak resource usage reproduction. The remaining 20% of tasks, however, exhibit a higher ERD ranging from 32 to 449, indicating a more urgent demand to reproduce their peak resource usage early on. A comparable trend is observed for memory resources, where a substantial proportion of tasks demonstrate a high ERD within the range of 0.015 to 0.399.

Guide for design: The workload generation should prioritizes the tasks with the high ERD to reproduce their maximal resource usage, thus alleviating the potential simulation deviations on the workload-level resource usage statistics.

4. Workload construction and generation

Work in this paper focuses on how to generate synthetic batch task workloads based on traces lacking explicit temporal information on task-level resource usage. The objective is to ensure the generated task workloads can not only accurately mirror their statistical resource usage characteristics extracted from the trace but also reproduce temporal resource usage patterns at the machine level. To this end, STWGEN is proposed in this paper. As shown in Fig. 5, there are two critical parts in STWGEN: workload building block construction and workload sequence generation. Workload building block construction aims to develop parameterized program units that can precisely generate the required resource usage based on parameter settings. With the absence of explicit task-level temporal resource usage data in the trace, workload sequence generation centers on reconstructing the resource usage time series formed during a batch task execution and taking the reconstructed sequence as a foundation to generate the complete workload program through assembling the pertinent building blocks. In addition, STWGEN incorporates a workload submission mechanism, enabling the practical replay of synthetic workloads reliant on information from trace data.

4.1. Construction of workload building blocks

In our work, a workload building block is defined as a customized C program segment that is capable of mimicing the desired resource usage during batch task execution. Based on observation #1 outlined in Section 3, we have designed workload building blocks to produce CPU usage and memory usage separately. To ensure accurate and efficient production of diverse resource usages, the designed building blocks must fulfill two prerequisites: they must possess lightweight computational logic, and their executions must be controllable. To achieve lightweightness, we opt for the most simple and straightforward operations as the components of the workload building blocks. To ensure controllability, we have designed the building blocks to be parameterized, allowing us to adjust their resource consumption through parameter settings.

4.1.1. Building block for CPU usage

This module is designed to simulate the batch task's CPU resource consumption. Its primary function is to use loop sum calculations to mimic CPU resource usage. As a computationally intensive operation, the sum operation can maximize the utilization of CPU resources. To generate varying CPU resource utilization, we have integrated sleep operations within the loop body to simulate idle states of CPU, thereby adjusting the amount of CPU resource usage during a specific time period. The number of loop iterations and the sleep durations are set as the parameters of this building block. In the current implementation, the parameters are adjusted on a one-second interval. The duration of the sleep operation for a one-second period is determined based on the targeted CPU utilization during that period. For instance, if the desired CPU utilization is 70%, the sleep duration would be set to 0.3 s. By tuning these parameters, we can precisely control the synthetic workload's CPU usage at any time point during execution. Furthermore, if a batch task's CPU usage surpasses the capacity of a single core (i.e., exceeds 100%), this building block can automatically create multiple threads, with each thread executing the same loop operation and sharing the sleep duration setting. The accumulation of CPU usage across multiple threads can accurately simulate the CPU usage of the task.

4.1.2. Building block for memory usage

This module is responsible for simulating batch task's memory consumption recorded in trace data. This building block consists of a loop program. Echoing Observation #3 outlined in Section 3, we adopt GLIBC memory [35] management functions within the loop body to simulate the occupation of various amounts of memory space during the task execution. The loop iteration is executed every one second.

In particular, in the first loop iteration, the *malloc* and *memset* function is called to allocate the desired amount of memory space, referred

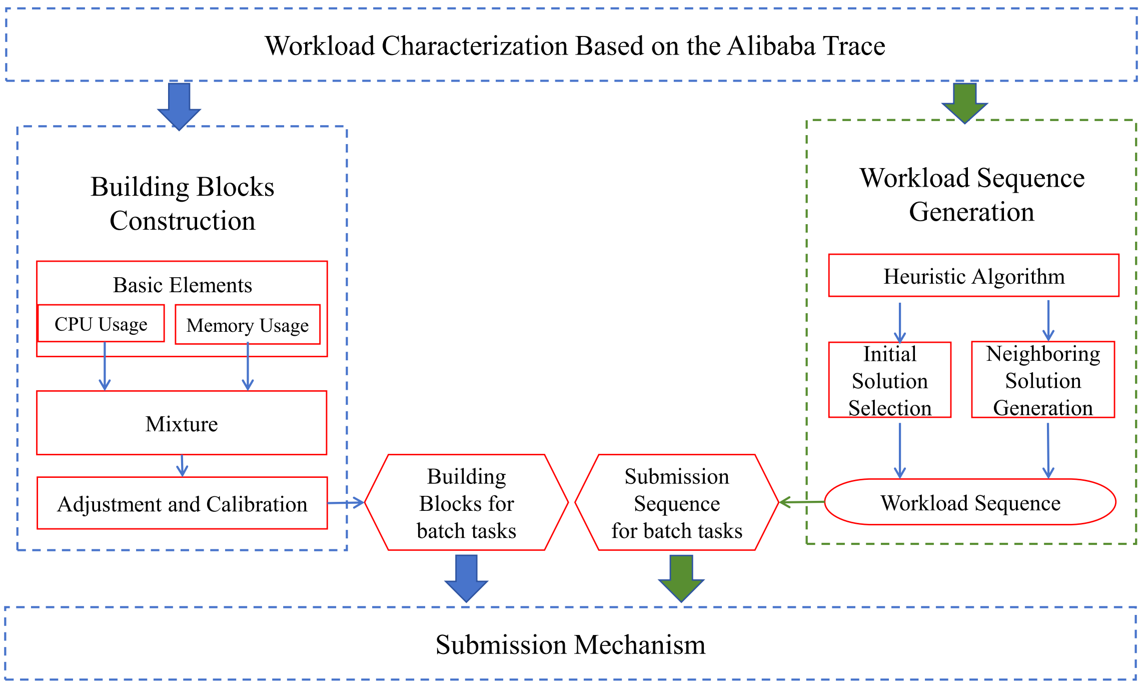

Workload Characterization Based on the Alibaba Trace
Building Blocks Construction
Basic Elements
CPU Usage
Memory Usage
Mixture
Adjustment and Calibration
Building Blocks for batch tasks
Submission Sequence for batch tasks
Workload Sequence Generation
Heuristic Algorithm
Initial Solution Selection
Neighboring Solution Generation
Workload Sequence
Submission Mechanism

Fig. 5. Framework of STWGEN.

to as the initial memory allocation. In the subsequent loop iterations, the allocated memory space is adjusted/resized with *realloc* function. If the desired amount of allocated memory is less than that in previous iteration, *realloc* function simply truncate the already allocated memory space. Conversely, if it is greater, the *memset* function is called to set values for the newly allocated space. The adoption of *realloc* function stems from Observation #3 in Section 3. Since the memory usage of tasks undergoes minor changes during their execution, *realloc* function effectively makes slight adjustments to the existing allocated memory space to simulate such changes. This approach eliminates the time overhead and the extra CPU consumption associated with frequently allocating and releasing memory across loop iterations using *malloc* and free functions, thereby enhancing the efficiency of memory usage reproduction and avoiding the bias in CPU usage simulation.

Furthermore, operations like *malloc* and *realloc* introduce additional memory space overhead due to metadata recording, heap and stack occupation. This overhead becomes particularly significant when the required memory usage is comparatively small, such as in tasks from Alibaba traces that utilize approximately 3% of the total machine memory. This results in a considerable deviation between the memory usage generated by our building block and the actual required usage. To address this issue, we sample overhead data by invoking the *malloc* and *realloc* functions with varying amounts of required memory allocation and reallocation. Subsequently, we utilize these sampled data to construct a linear overhead prediction function. This function aids us in precisely adjusting the building block parameters, thereby ensuring that the resulting memory resource usage aligns precisely with our expectations.

4.2. *Workload sequence generation*

We describe the reconstruction method for batch task-level resource usage sequence in this section and propose the mechanism to generate the complete workload program in section 4.3.

The goal of sequence reconstruction is to restore resource usage at each time point throughout the execution of a batch task. This reconstruction must guarantee that the statistical properties of resource usage, specifically the average and maximum values, for a synthetic task workload are in alignment with the corresponding information extracted from the trace data. Additionally, it is crucial to ensure that the aggregated resource usage of concurrent synthetic tasks running on a specific machine accurately reflects the overall resource usage observed on that machine in the trace data. We thus formulate the resource usage sequence reconstruction issue as a multi-objective optimization problem and adopt a heuristic algorithm to solve it.

We adopt the Simulated Annealing (SA) heuristic to find the optimized solution for resource sequence reconstruction. SA heuristic algorithm mimics the physical annealing process to search for optimal solutions to complex problems [36,37].

When designing SA for a specific application, four key points should be figured out: the cost function definition, the initial solution selection, the neighboring solution generation, and the acceptance criterion determination. Algorithm 1 proposes a detailed framework for batch task resource sequence reconstruction using Simulated Annealing (SA). It commences with an initial workload resource sequence solution and an initial temperature T_0, and then iteratively explores the search space while gradually reducing the temperature according to a cooling schedule. In each iteration, a new candidate solution (neighbor) is generated from the current solution, and its acceptance is determined in a probabilistic manner based on a specified acceptance criterion.

The cost function in SA algorithm serves to evaluate the quality of each candidate solution during the optimization process. In our design, the cost function is defined based on Eqs. (1) (2):

$$CostFunction = DIV_{task} + DIV_{machine} \tag{4}$$

In our work, the resource usage sequence reconstruction for CPU and memory resources is conducted separately. In each instance, the reconstruction deviation is solely calculated based on the target resource. In addition, the task-level deviation DIV_{task} primarily comprises two components: deviations in both average and maximum resource usage statistics. We incorporate these deviations linearly in the cost function. Evidently, this cost function takes into account both the machine-level and task-level resource sequence reconstruction errors, thereby effectively guiding the algorithm towards finding an optimal solution. The acceptance criterion is used to determine whether a newly generated solution should be accepted or rejected. We adopt the Metropolis

Algorithm 1: SA-Based Workload Sequence Generation

```
Input  : Batch task resource usage statistic list Task_list, Time
         period TP, Machine resource usage sequence
         Machine_seq, Initial temperature $T_0$, Cooling rate CR,
         Temperature threshold min_T
Output: Optimal solution for batch tasks' resource usage
         sequences Solution
1  Generate initial solutions for all batch tasks as init_solution.
2  Set current_solution as init_solution.
3  Set Current Temperature $T$ as $T_0$.
4  while $T > min\_T$ do
5      Calculate $P_{acceptance}$ as the acceptance probability.
6      Calculate current_cost for current_solution.
7      if $random() > 0.8$ then
8          generate neighbor_solution by naive search method.
9      else
10         generate neighbor_solution by JAYA algorithm with input
             of $P_{\mathrm{acceptance}}$.
11     end
12     Calculate neighbor_solution's cost as neighbor_cost.
13     if neighbor_cost < current_cost then
14         set current_solution as neighbor_solution.
15     else
16         if random() > $P_{acceptance}(T)$ then
17             set current_solution as neighbor_solution.
18         end
19     end
20     Decrease temperature $T$ by $CR$.
21 end
22 Return Solution = current_solution.
```

criterion in our design which can be described as follows: if the cost of the neighboring solution (that is, the newly generated solution) is less than that of the current solution, it will be accepted. Otherwise, it will be accepted with the probability of $P_{acceptance}$. $P_{acceptance}$ is calculated based on the current annealing temperature T.

$$P_{acceptance} = e^{\frac{neighbor\_cost-current\_cost}{T}} \tag{5}$$

The efficacy of the final resource usage sequence significantly depends on the quality of the initial solution and the methodology for generating neighboring solutions. We will delve into the specifics of these aspects in the subsequent sections.

4.2.1. *Initial solution selection*

To enhance the reproducibility quality of the final solution, we should strive to select an initial resource usage sequence for a batch task that aligns with its corresponding statistics, particularly in terms of the average and maximal resource usage recorded in trace data. This selection must adhere to the constraint of the machine-level total resource utilization.

Based on observation #4 in Section 3, the initial solution selection algorithm is outlined in Algorithm 2. For each task, a preliminary resource usage sequence is generated, following a normal distribution, and derived from its statistical average (line #2). This preliminary sequence constitutes the initial step towards the creation of the final initial solution. Initially, all tasks are designated as "unallocated", indicating that they have not yet reached their maximum resource usage during any point of execution (line#3). Transitioning a workload to the "allocated" state signifies that it has achieved its maximum resource usage at a specific moment during its execution time.

By sequentially traversing each moment in time over the given period, concurrent tasks executing at a specific time point are sorted in descending order based on their Early Reproduction Demand (ERD), as defined in observation #4 (Line #8). These tasks are then examined sequentially. If a task remains in the "unallocated" state and the machine's remaining resource usage quotas is still sufficient, it is marked as "allocated", and its resource usage is set to its maximum value for that specific sampling point (Line #9 -#12). If not, the workload is allocated the remaining resource usage quotas available on the machine (line #14 - #15).

By utilizing an initial sequence generation based on average resource usage, the chosen initial solution for a batch task has a high likelihood of aligning with its corresponding average statistic. Furthermore, by employing ERD as the task priority, the proposed method ensures that tasks with higher maximum resource usage statistics and shorter execution durations are prioritized for reproducing their peak resource usage.

Algorithm 2: Initial Workload Sequence Solution Selection

```
Input  : Batch task resource usage statistic list Task_list, Time
         period TP, Machine resource usage sequence
         Machine_seq.
Output: Initial solution for batch tasks' resource usage
         sequences Solution
1  for $task_j$ in $Task\_list$ do
2      set $Solution[task_j]$ for $task_j$ following normal distribution($\mu$
         is the average resource usage of $task_j$).
3      Calculate ERD of $task_j$ and set $task_j$ state as 'unallocated'.
4  end
5  for $timestamp_i$ in $TP$ do
6      Set $Resource\_left$ as the total resource usage at $timestamp_i$ in
         $Machine\_seq$.
7      while $Resource\_left > 0$ do
8          Find the concurrent task at $timestamp_i$ with the maximal
             ERD and 'unallocated' state as $Task_{Urg}$.
9          if $Resource\_left >$ peak resource usage of $Task_{Urg}$ then
10             Set $Solution[Task_{Urg}]$ at $timestamp_i$ as the peak
                 resource usage of $Task_{Urg}$.
11             Decrease $Resource\_left$ by the peak resource usage of
                 $Task_{Urg}$.
12             Mark $Task_{Urg}$ state as 'allocated'.
13         else
14             Set $Solution[Task_{Urg}]$ at $timestamp_i$ as $Resource\_left$.
15             Set $Resource\_left$ to 0.
16         end
17     end
18 end
19 return $Solution$.
```

4.2.2. *Neighboring solution generation*

Neighboring solution generation plays a crucial role in enhancing the diversity of solutions and preventing simulated annealing from falling into the local optima. In our design, two strategies are randomly employed in Neighboring solution generation:

- Naive Search Strategy: Based on the current solution, a task's solution sequence and a position within that sequence are randomly selected. A random floating-point number within the range of $[-5, 5]$ is then added to the resource usage value on this selected position, resulting in a new neighboring solution.
- Strategy based on JAYA Algorithm: The Jaya algorithm [38] is a meta-heuristic optimization technique that operates on the principle of continuous improvement by moving towards better solutions and away from poorer ones. Specific steps of this strategy are as follows:
  1) Randomly select a task's solution sequence and name it $Solution_{Cur}$.

2) Add a standard normal distribution random value at each position of $Solution_{Cur}$. Repeat this operation 10 times to generate 10 candidate solutions.
(3) Calculate the cost values of the ten generated solutions and find the solution with the minimum and maximum cost values as $Solution_{Best}$ and $Solution_{Worst}$.
(4) Use JAYA algorithm update formula to generate $Solution_{New}$ by $Solution_{Cur}$, making it get close to the $Solution_{Best}$ and staying away from $Solution_{Worst}$.
5) If the cost value of $Solution_{New}$ is less than that of $Solution_{Cur}$, updating $Solution_{Cur}$ as $Solution_{New}$. Otherwise, Combine with the Simulated Annealing algorithm to accept $Solution_{New}$ with a certain probability.
6) Repeat step (2)–(5), until the designated iteration number is reached, and choose $Solution_{Cur}$ as the final solution.

The naive search strategy commences with our carefully crafted optimized initial solution and proceeds by making incremental adjustments along the search trajectory. This strategy ensures the exploitation capability of our neighboring solution generation. While the Jaya-based strategy serves as a complement to enhance the exploration capability of our neighboring solution generation. It strives to find beneficial solutions by exploring new regions of the search space while simultaneously avoiding solutions that are less promising. In addition, the design of jaya-based search is intimately linked to the principles of simulated annealing heuristic. As the rounds of simulated annealing progress, this search's tendency to explore new solutions diminishes, instead focusing more on the exploitation and optimization of existing solutions. In summary, by combining these two strategies, our neighboring solution generation method ensures to output the high-quality solution. Experimental results on Alibaba trace prove that adopting Jaya-based neighboring solution searching helps to reduce the deviation on workload's maximal CPU resource usage by 34.2%.

4.3. Workload submission mechanism

We have implemented a workload submission mechanism that enables the authentic generation of runnable batch task workloads, leveraging our proposed workload construction and generation methods.

This submission mechanism involves creating a Python script that prompts users to specify a machine within the trace and the corresponding time period for workload replay. Subsequently, the script reconstructs the resource usage sequence for the designated batch tasks within the trace. The synthetic workloads are then launched by executing the CPU and memory building blocks as concurrent threads, which take the generated resource usage sequence as input and replicate the resource usage pattern every second.

5. Performance evaluation

We introduce the experimental setup, followed by a detailed presentation of the evaluation results in this section. Our evaluation primarily comprises two parts: first, we evaluate the resource usage reproduction accuracy of STWGEN; second, we compare the performance of STWGEN with state-of-the-art trace-based workload generation methods.

5.1. Experimental setup

From Alibaba cluster data 2018, we choose all task instances running on three representative machines: m_1935, m_1983 and m_1940, specifically on the fourth day within the 8-day trace recording period to evaluate STWGEN. Among the three selected machines, m_1935 has the high average and variance of resource usage, while m_1983 and m_1940 respectively have the medium and low levels during the selected time period. The task-level resource usage statistics are derived from *batch_instance* table and the machine-level temporal resource usage data is obtained by subtracting the resource usage data in *container_usage* table from the corresponding resource usage data from *machine_usage* table, based on aligned timestamps. The experimental environment was configured on Alibaba Cloud using Linux 3.2104 LTS 64-bit OS, with the system specifications set to 'ecs.hfc6.16xlarge', featuring 64 vCPUs and 128 GB of memory.

5.2. Metrics

According to the definition in Eqs. (1)(2), we formulate three metrics to evaluate the workload and machine-level resource usage reproduction accuracy achieved by STWGEN and other baselines.

$$DR_{Machine}(i) = \frac{|Sum(CT_i) - uc_i|}{uc_i} \tag{6}$$

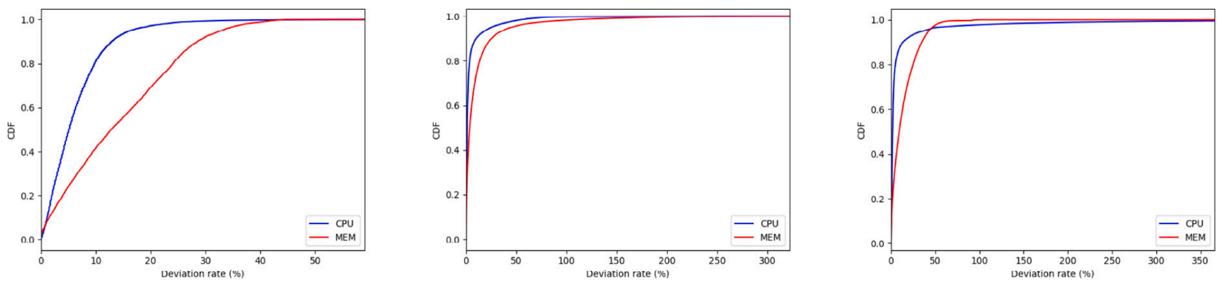
$$DR_{Task\_max}(j) = \frac{|Max(rs_j) - max_j|}{max_j} \tag{7}$$

$$DR_{Task\_avg}(j) = \frac{|Avg(rs_j) - avg_j|}{avg_j} \tag{8}$$

$DR_{Machine}(i)$ represents the deviation rate in the reproduction of machine-level resource usage, where, $Sum(\cdot)$ function signifies the accumulated resource usage produced with the generated concurrent batch tasks at a specific time point i on a machine. uv_i stands for the observed total resource usage of the machine at the corresponding time point in the trace.$DR_{Task\_max}(j)$ and $DR_{Task\_avg}(j)$ represent the deviation rates in the reproduction of the maximal and average resource usage statistics at task level, respectively. Where, $Max(\cdot)$ and $Avg(\cdot)$ functions signify the statistical maximum and average values, respectively, of the resource usage sequence generated by the synthetic workload for task j. max_j and avg_j refer to the corresponding statistics recorded in the trace. We use the aforementioned metrics for CPU and memory resources, respectively.

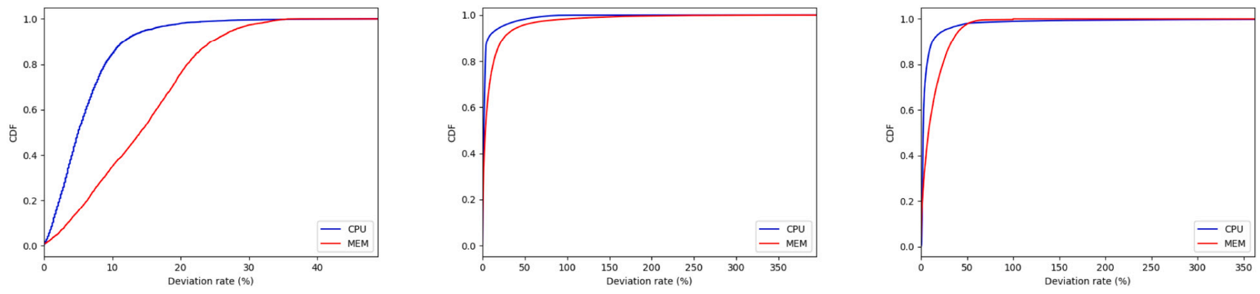
5.3. Resource usage reproduction accuracy with STWGEN

Figs. 6, 7, and 8 illustrate the distributions of deviation rates in the reproduction of resource usage with STWGEN on machines m_1935, m_1983, and m_1940, respectively. At the machine level, the average deviation rate for CPU usage across all recorded time points on a specific machine is under 6.4%, whereas for memory usage, it is less than 14.3%. On the task level, the average deviation rate for the maximum and average statistics of CPU usage among all batch tasks is below 5.4% and 9.6%, respectively. Similarly, the average deviation rate for the maximum and average statistics of memory usage is less than 11.4% and 14.1%. Overall, the reproduction performance on CPU resource usage surpasses that on memory resource usage. For instance, on each of the selected machines, over 85% of the recorded time points show deviation rates for CPU usage that are below 15%, whereas only 55% of the time points achieve such a deviation rate for memory usage. This is due to that most batch tasks in Alibaba trace have 1 small amount of memory usage, making them more sensitive to slight biases in resource usage reproduction, which ultimately results in higher deviation rates. However, given the low memory resource usage recorded in Alibaba trace, even a deviation rate of 32.67% (exceeding those on the 95th percentile on all these machines) translates into a mere absolute deviation of 0.67, which is negligible when compared to the memory usage range of [0, 100]. Consequently, we deem the maximum average deviation rate of 14.3% achieved by STWGEN in memory resource usage reproduction as acceptable. In addition, among the three selected machines, tasks on m_1940 achieve the best reproduction performance. This is because the variation in total resource usage on m_1940 is less than that of the other two machines, enabling the workload building blocks to adjust their CPU/memory usage less frequently and reducing the search cost of finding optimized reconstructed resource usage sequences for batch tasks.



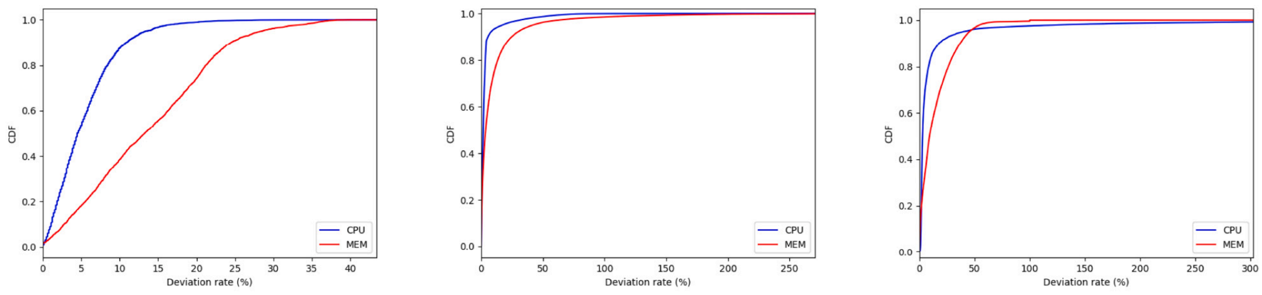
(a) $DR_{Machine}$ (b) DR_{Task_max} (c) DR_{Task_avg}

Fig. 6. Distribution of deviation rates on CPU and memory usage reproductions on m_1935.



(a) $DR_{Machine}$ (b) DR_{Task_max} (c) DR_{Task_avg}

Fig. 7. Distribution of deviation rates on CPU and memory usage reproductions on m_1983.



(a) $DR_{Machine}$ (b) DR_{Task_max} (c) DR_{Task_avg}

Fig. 8. Distribution of deviation rates on CPU and memory usage reproductions on m_1940.

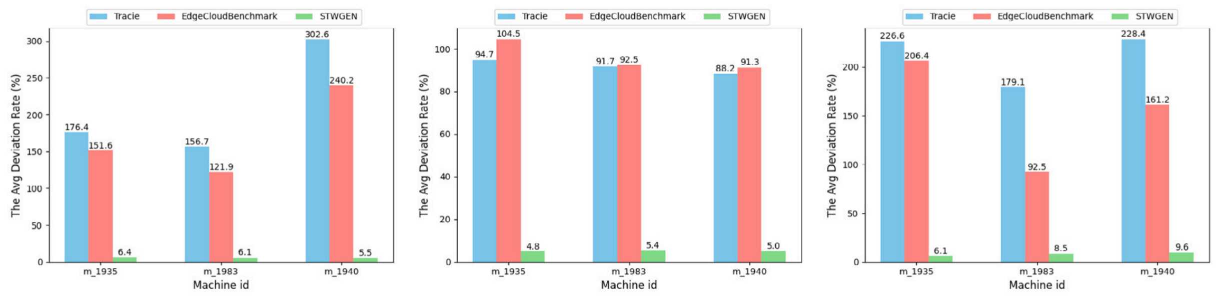
5.4. Comparison to the state-of-art methods

To verify the superiority of STWGEN, we compare it with two state-of-art methods: Tracie [25] and EdgeCloudBenchmark [24].

Tracie utilizes Parametric Density Estimation (PDE) to derive the probability distribution function (PDF) of batch tasks' resource utilization characteristics. Following this, it randomly generates resource usage sequences for batch tasks that adhere to the derived probability distributions. Subsequently, Tracie selects applications from the TPC benchmark suite([27]) and the Rodinia benchmark([39]) that exhibit comparable resource utilization statistics, using them as building blocks to craft synthetic workloads. EdgeCloudBenchmark employs a clustering technique to categorize the batch tasks recorded in the trace data into several groups, where tasks within the same group exhibit similar resource usage statistics. It then randomly selects a task from each group to represent the resource usage characteristics of the tasks within that group. It utilized Apache Bench test tool as build blocks to generate the synthetic workloads.

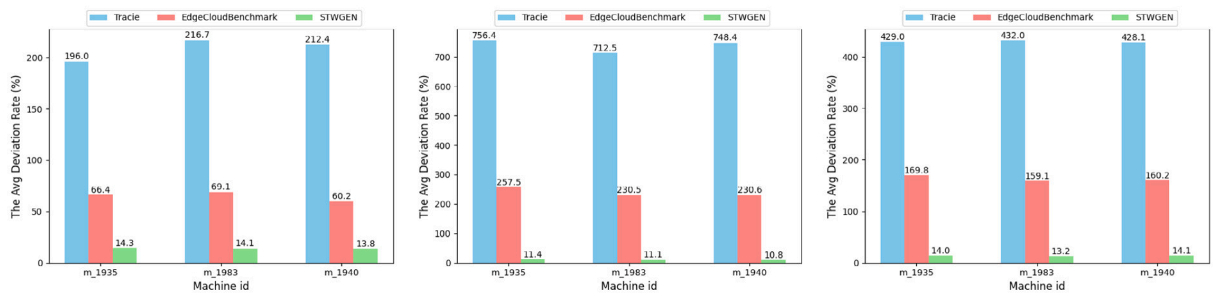
We conduct the experiments on machines m_1935, m_1983, and m_1940, and the results are presented in Figs. 9 and 10. STWGEN exhibits the most stable performance on all three machines, achieving the lowest average deviation rate of machine-level resource usage reproduction, which stands below 14.3%, and the lowest average deviation rate of task-level resource usage statistics reproduction is less than 14.1%. In contrast, Tracie and EdgeCloudBenchmark show significantly higher deviation rates in terms of resource usage reproductions. Specifically, Tracie demonstrates an average deviation rate of up to 302.6% for machine-level resource usage across the three machines, while the average deviation rate for task-level maximum and average resource usage statistics reached to 756.4% and 432%, respectively. The corresponding rates of EdgeCloudBenchmark are 240.2%, 257.5%, and 206.4%, respectively. Overall, compared to the two baselines, STWGEN can reduce the deviation rates of the resource usage reproductions by up to 98.6% and a minimum of 77.1%.

The superiority of STWGEN lies in two aspects. Firstly, it introduces lightweight and agile building blocks for task-level workload generation. Unlike the predefined benchmark applications used in Tracie, which generate limited and relatively fixed temporal resource utilization patterns, our proposed building blocks can dynamically generate resource consumption profiles on demand, accommodating the diverse resource utilization signatures of batch tasks in large-scale data centers. Secondly, using the proposed heuristic algorithm, STWGEN can finely



(a) $DR_{Machine}$ (b) DR_{Task_max} (c) DR_{Task_avg}

Fig. 9. Performance comparison on CPU usage reproduction.



(a) $DR_{Machine}$ (b) DR_{Task_max} (c) DR_{Task_avg}

Fig. 10. Performance comparison on memory usage reproduction.

reconstruct the resource usage of individual batch tasks, considering both their resource usage statistics and machine-level resource constraints. Such time point-wise machine-level constraints, contributes to the accurate reproduction of a task's resource usage at any point during its runtime. However, Tracie and EdgeCloudBenchmark fail to incorporate workload-level coarse-grained and machine-level fine-grained resource usage characteristics. They merely characterize cluster-level task resource usage statistics and use these statistics to randomly generate synthetic workloads, thus lacking the precision to mimic individual tasks accurately.

6. Related works

Traces from large-scale production cloud platforms are pivotal in the workload analysis and generation. Three prominent traces commonly used for cloud workload generation are Google ClusterData2011 [18], Azure PublicDataset2017 [40,41], and Alibaba Clusterdata2018 [19]. Google ClusterData2011 comprises extensive data on hundreds of thousands of job-task-structured workloads, particularly the detailed resource usage tracked over time for each task and their micro-architecture behavior information. Azure PublicDataset2017 represents the workload of virtual machines (VMs) within Microsoft Azure, collected in 2017. It documents the resource usage of approximately 2 million monitored VMs, recorded every 5 min, resulting in comprehensive time series data. Among these three traces, Alibaba Clusterdata2018 stands out as the most recent. It records resource usage information for over 12 million batch tasks. Distinct from the other two traces, Alibaba Clusterdata2018 simply captures statistical summaries for batch tasks' resource usage, encompassing metrics like maximum and average resource usage during task execution.

From the perspective of reproducing resource usage characteristics, trace-based workload generation in data centers can be divided into four levels: cluster-level, virtual machine-level, job-level, and task-level. Cluster- and VM-level workload generations replicate overall resource usage patterns of cloud clusters or individual VMs [23,26,42] [29,43,44], but not individual workloads. Hence, they excel in data center capacity planning, but lack precision in fine-grained resource scheduling. Job-level generation aims to simulate the characteristics of job workloads, such as the resource usage pattern [45], the job structure [45], the task size and the execution duration of jobs [22] . As the basic unit for cloud workload submission, Job-level workload generation primarily targets workloads that meet average resource usage and expected duration for jobs, but often neglects precise replication of resource usage for individual tasks within a job. Tasks serve as the fundamental units for cloud workload execution and resource usage [46]. Task-level workload generation aims to accurately replicate resource usage characteristics from trace data, crucial for effective resource scheduling and workload migration [24,25,28]. This paper focuses on trace-based batch task workload generation in cloud data centers.

From the technical view, trace-based workload generation can be categorized into two types based on the granularity of reproducing resource usage characteristics: coarse-grained and fine-grained generation. Coarse-grained workload generation involves extracting statistical features of resource utilization from trace data, such as average and peak resource usage, and synthesizing a comprehensive workload that mirrors these characteristics using a combination of typical benchmark applications as building blocks. For instance, Sfakianakis et al. [25] employ techniques like Parametric Density Estimation (PDE), Non-parametric Density Estimation (NDE), or Kernel Density Estimation (KDE) to determine the probability distribution function (PDF) of resource utilization characteristics in Google trace batch tasks. Subsequently, it selects applications from the TPC benchmark suite [27] and Rodinia benchmark [39] that exhibit comparable resource utilization statistics to craft a holistic workload. Additionally, Koltuk et al. [23] conduct a comprehensive analysis of resource utilization distribution and structural attributes of batch jobs in Alibaba trace, leveraging mapreduce tasks within BDGS to create synthetic workloads for simulation and analysis purposes. Furthermore, Wen et al. [24] apply k-means clustering to categorize batch tasks in Alibaba trace, randomly selecting representative tasks from each cluster. A distributed framework based on microservices is then utilized to simulate the concurrent execution of these tasks. Lastly, the Apache Bench test tool is used to generate resource consumption patterns for each individual task, ensuring alignment with the characteristics of the selected representative tasks. Although commonly employed, coarse-grained methods solely capture statistical resource usage, neglecting intricate temporal patterns, thereby reducing workload reproduction precision. Benchmark applications, while representative, still differ from real-world workloads in the temporal resource usage patterns, and this discrepancy widens when multiple applications are combined to generate the synthetic workloads.

Fine-grained workload generation involves constructing synthesized workloads that precisely match the temporal patterns of resource usage recorded in trace data. In contrast to coarse-grained generation, fine-grained methods strive to accurately replicate the resource utilization at every instant during the workload's execution, encompassing not just statistical features but also the dynamic behavior. Using RWB (Reducible Workload Block) as the building block for workload generation, Han et al. [28] combine various RWBs to synthesize workload for each task moment, drawing on the micro-architectural behavior characteristics of tasks captured in Google trace. Koltuk et al. [23,42] aim to replicate virtual machine workload resource usage based on Azure trace. It identifies a cumulative distribution function fitting the trace samples and based on the periodic characteristics in VM's resource usage, it adjusts resource usage to align with the distribution while minimizing auto-correlation. Lin et al. [47] employ Generative Adversarial Networks to generate the time-dependent cloud workload, which does not require any prior knowledge to do distribution analysis. However, the article fails to provide specific details about the building blocks used for workload generation, rendering the application of this method to the construction of authentic synthetic workloads challenging. In summary, while fine-grained load generation methods excel at generating intricate sequences of resource usage, they typically depend on explicit temporal information extracted from traces to accurately reconstruct resource usage characteristics at the workload or machine level. Nevertheless, this approach presents difficulties when applied to large-scale traces with limited temporal information, such as the Alibaba trace. Additionally, existing fine-grained methods tailored for batch tasks rely heavily on assembly instructions, which restricts their portability and renders them unusable for traces lacking micro-architectural metric information. Consequently, the challenge remains of developing an efficient method for fine-grained batch task workload generation that can handle large-scale traces lacking both temporal information on resource usage and detailed system-level behavior data.

7. Conclusion

In light of the challenge posed by the absence of precise temporal information of workload-level resource usage in modern data center traces, this paper proposes STWGEN, a novel trace-based batch task workload generation method based on the statistic-based Alibaba trace. STWGEN accurately reproduces the batch task's resource usage sequence by incorporating the task-level coarse-grained resource statistics and the machine-level fine-grained resource constraints. The extensive experimental results affirm the superiority of STWGEN method to the state-of-art baselines. In the future, our work will be extended to consider task dependencies in the workload generation.

CRediT authorship contribution statement

Yi Liang: Writing – review & editing, Supervision, Methodology. **Nianyi Ruan:** Writing – original draft, Validation, Software, Resources, Investigation. **Lan Yi:** Methodology, Writing – original draft, Writing – review & editing. **Xing Su:** Writing – original draft, Writing – review & editing.

Declaration of competing 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.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62276011.

References

[1] L.A. Barroso, U. Hölzle, P. Ranganathan, The Datacenter as a Computer: Designing Warehouse-Scale Machines, Springer Nature, 2019.
[2] G. Wang, T.E. Ng, The impact of virtualization on network performance of amazon ec2 data center, in: IEEE INFOCOM, 2010, pp. 1–9.
[3] K. Kant, Data center evolution: A tutorial on state of the art, issues, and challenges, Comput. Netw. 53 (17) (2009) 2939–2965.
[4] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, X. Fu, Recent advances of resource allocation in network function virtualization, IEEE Trans. Parallel Distrib. Syst. 32 (2) (2020) 295–314.
[5] Y. Huang, H. Xu, H. Gao, X. Ma, W. Hussain, SSUR: An approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center, IEEE Trans. Green Commun. Netw. 5 (2) (2021) 670–681.
[6] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826–831.
[7] A. Chandra, W. Gong, P. Shenoy, Dynamic resource allocation for shared data centers using online measurements, in: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2003, pp. 300–301.
[8] H. Raj, R. Nathuji, A. Singh, P. England, Resource management for isolation enhanced cloud services, in: ACM Workshop on Cloud Computing Security, 2009, pp. 77–84.
[9] D. Gmach, J. Rolia, L. Cherkasova, A. Kemper, Capacity management and demand prediction for next generation data centers, in: IEEE International Conference on Web Services, 2007, pp. 43–50.
[10] A. Gandhi, M. Harchol-Balter, R. Raghunathan, M.A. Kozuch, Autoscale: Dynamic, robust capacity management for multi-tier data centers, ACM Trans. Comput. Syst. 30 (4) (2012) 1–26.
[11] Y. Song, H. Wang, Y. Li, B. Feng, Y. Sun, Multi-tiered on-demand resource scheduling for VM-based data center, in: IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 148–155.
[12] L. Wang, J. Zhan, W. Shi, Y. Liang, In cloud, can scientific communities benefit from the economies of scale? IEEE Trans. Parallel Distrib. Syst. 23 (2) (2011) 296–303.
[13] A. Greenberg, J. Hamilton, D.A. Maltz, P. Patel, The cost of a cloud: Research problems in data center networks, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.
[14] R. Panda, L.K. John, Proxy benchmarks for emerging big-data workloads, in: International Conference on Parallel Architectures and Compilation Techniques, 2017, pp. 105–116.
[15] J. Moore, J. Chase, K. Farkas, P. Ranganathan, Data center workload monitoring, analysis, and emulation, in: Eighth Workshop on Computer Architecture Evaluation using Commercial Workloads, 2005, pp. 1–8.
[16] R. Han, L.K. John, J. Zhan, Benchmarking big data systems: A review, IEEE Trans. Serv. Comput. 11 (3) (2017) 580–597.
[17] R. Han, M.M. Ghanem, L. Guo, Y. Guo, M. Osmond, Enabling cost-aware and adaptive elasticity of multi-tier cloud applications, Future Gener. Comput. Syst. 32 (2014) 82–98.
[18] Google, The google cluster trace, https://github.com/google/cluster-data/.
[19] Alibaba, Alibaba/clusterdata, https://github.com/alibaba/clusterdata/.
[20] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, Y. Bao, Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces, in: International Symposium on Quality of Service, 2019, pp. 1–10.
[21] Z. Jia, J. Zhan, L. Wang, R. Han, S.A. McKee, Q. Yang, C. Luo, J. Li, Characterizing and subsetting big data workloads, in: IEEE International Symposium on Workload Characterization, 2014, pp. 191–201.
[22] J. Zhu, B. Lu, X. Yu, J. Xu, T. Wo, An approach to workload generation for cloud benchmarking: A view from alibaba trace, in: IEEE 15th International Symposium on Autonomous Decentralized System, 2023, pp. 1–8.
[23] F. Koltuk, A. Yazar, E.G. Schmidt, Cloudgen: Workload generation for the evaluation of cloud computing systems, in: Signal Processing and Communications Applications Conference, 2019, pp. 1–4.
[24] S. Wen, H. Deng, K. Qiu, R. Han, EdgeCloudBenchmark: A benchmark driven by real trace to generate cloud-edge workloads, in: IEEE International Conference on Sensing, Diagnostics, Prognostics, and Control, 2022, pp. 377–382.

[25] Y. Sfakianakis, E. Kanellou, M. Marazakis, A. Bilas, Trace-based workload generation and execution, in: International Conference on Parallel and Distributed Computing, 2021, pp. 37–54.
[26] P. Jacquet, T. Ledoux, R. Rouvoy, Cloudfactory: An open toolkit to generate production-like workloads for cloud infrastructures, in: IEEE International Conference on Cloud Engineering, 2023, pp. 81–91.
[27] T.P.P. Council, Transaction processing performance council, http://www.tpc.org.
[28] R. Han, Z. Zong, F. Zhang, J.L. Vazquez-Poletti, Z. Jia, L. Wang, Cloudmix: Generating diverse and reducible workloads for cloud systems, in: IEEE 10th International Conference on Cloud Computing, 2017, pp. 496–503.
[29] R. Han, S. Zhan, C. Shao, J. Wang, L.K. John, J. Xu, G. Lu, L. Wang, Bigdatabench-mt: A benchmark tool for generating realistic mixed data center workloads, in: Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 6th Workshop, 2016, pp. 10–21.
[30] J. Chen, Y. Zhang, X. Jiang, L. Zhao, Z. Cao, Q. Liu, DWT: Decoupled workload tracing for data centers, in: IEEE International Symposium on High Performance Computer Architecture, 2020, pp. 677–688.
[31] N.M. Josuttis, The C++ Standard Library: A Tutorial and Reference, Addison-Wesley, 2012.
[32] W. Chen, K. Ye, Y. Wang, G. Xu, C.-Z. Xu, How does the workload look like in production cloud? Analysis and clustering of workloads on alibaba cluster trace, in: IEEE 24th International Conference on Parallel and Distributed Systems, 2018, pp. 102–109.
[33] C. Jiang, Y. Qiu, W. Shi, Z. Ge, J. Wang, S. Chen, C. Cérin, Z. Ren, G. Xu, J. Lin, Characterizing co-located workloads in alibaba cloud datacenters, IEEE Trans. Cloud Comput. 10 (4) (2020) 2381–2397.
[34] P.C. Coefficient, Pearson's correlation coefficient, New Zealand Med. J. 109 (1015) (1996) 38.
[35] W. von Hagen, Building and installing glibc, in: The Definitive Guide to GCC, Springer, 2006, pp. 247–279.
[36] X. Cao, G. Li, Q. Ye, R. Zhou, G. Ma, F. Zhou, Multi-objective optimization of permanent magnet synchronous motor based on elite retention hybrid simulated annealing algorithm, in: IEEE Conference on Industrial Electronics and Applications, 2017, pp. 535–540.
[37] Y. Hu, Y. Zuo, Z. Sun, Combination of simulated annealing algorithm and minimum horizontal line algorithm to solve two-dimensional pallet loading problem, in: Winter Simulation Conference, 2022, pp. 1956–1966.
[38] R. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, Int. J. Ind. Eng. Comput. 7 (1) (2016) 19–34.
[39] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, S.-H. Lee, K. Skadron, Rodinia: A benchmark suite for heterogeneous computing, in: IEEE International Symposium on Workload Characterization, 2009, pp. 44–54.
[40] Microsoft, Microsoft/Azure traces, https://github.com/Azure/AzurePublicDataset.
[41] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, R. Bianchini, Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms, in: Symposium on Operating Systems Principles, 2017, pp. 153–167.
[42] F. Koltuk, E.G. Schmidt, A novel method for the synthetic generation of non-iid workloads for cloud data centers, in: IEEE Symposium on Computers and Communications, 2020, pp. 1–6.
[43] S. Bergsma, T. Zeyl, A. Senderovich, J.C. Beck, Generating complex, realistic cloud workloads using recurrent neural networks, in: ACM SIGOPS 28th Symposium on Operating Systems Principles, 2021, pp. 376–391.
[44] J. Xu, J. Liu, J. Yao, A. Ma, L. Xu, X. Zhao, Conditional generative adversarial network based workload generation for cloud cluster, in: International Conference on Algorithms, Computing and Artificial Intelligence, 2022, pp. 1–6.
[45] Y. Liang, K. Chen, L. Yi, X. Su, X. Jin, DeGTeC: A deep graph-temporal clustering framework for data-parallel job characterization in data centers, Future Gener. Comput. Syst. 141 (2023) 81–95.
[46] P. Minet, E. Renault, I. Khoufi, S. Boumerdassi, Analyzing traces from a google data center, in: International Wireless Communications & Mobile Computing Conference, 2018, pp. 1167–1172.
[47] W. Lin, K. Yao, L. Zeng, F. Liu, C. Shan, X. Hong, A GAN-based method for time-dependent cloud workload generation, J. Parallel Distrib. Comput. 168 (2022) 33–44.