Dynamic estimation and Grid partitioning approach for Multi-Objective Optimization Problems in medical cloud federations

Trung-Dung Le, Verena Kantere, Laurent d'Orazio

Dynamic estimation and Grid partitioning approach for Multi-Objective Optimization Problems in medical cloud federations

Trung-Dung Le[1][0000−0001−9560−1180], Verena Kantere[2][0000−0002−3586−9406], and Laurent d'Orazio[1][0000−0001−8614−1848]

[1] Univ Rennes, 2 rue du Thabor - CS 46510 - 35065 Rennes CEDEX
firstname.lastname@irisa.fr
[2] University of Ottawa, 75 Laurier Ave E, Ottawa, ON K1N 6N5, Canada
vkantere@uOttawa.ca

Abstract. Data sharing is important in the medical domain. Sharing data allows large-scale analysis with many data sources to provide more accurate results. Cloud federations can leverage sharing medical data stored in different cloud platforms, such as Amazon, Microsoft, etc. The pay-as-you-go model in cloud federations raises important issues of Multi-Objective Optimization Problems (MOOP) related to users' preferences, such as response time, money, etc. However, optimizing a query in a cloud federation is complex with increasing the variety, especially due to a wide range of communications and pricing models. The variety of virtual machines configuration also leverages the high complexity in generating the space of candidate solutions. Indeed, in such a context, it is difficult to provide accurate estimations and optimal solutions to make relevant decisions. The first challenge is how to estimate accurate parameter values for MOOPs in a cloud federation consisting of different sites. To address the accurate estimation of parameter values problem, we present the Dynamic Regression Algorithm (DREAM). DREAM focuses on reducing the size of historical data while maintaining the estimation accuracy. The second challenge is how to find an approximate optimal solution in MOOPs using an efficient Multi-Objective Optimization algorithm. To address the problem of finding an approximate optimal solution, we present Non-dominated Sorting Genetic Algorithms based on Grid partitioning (NSGA-G) for MOOPs. The proposed algorithm is integrated into the Intelligent Resource Scheduler, a solution for heterogeneous databases, to solve MOOP in cloud federations. We validate our algorithms with experiments on a decision support benchmark.

Keywords: Cloud computing · Multiple Linear Regression · Cloud federations · Genetic Algorithm · Non-dominated Sorting Genetic Algorithm.

1 Introduction

Cloud federation is a paradigm of interconnecting the cloud environments of more than one service providers for multiple purposes of commercial, service quality, and user's requirements [35]. Besides of vendor lock-in and provider integration, a cloud federation has several types of heterogeneity and variability in the cloud environment, such as wide-range communications and pricing models.

Cloud federations can be seen as a major progress in cloud computing, in particular for the medical domain. Indeed, sharing medical data would improve healthcare. Federating resources makes it possible to access any information even on distributed hospital data on several sites. Besides, it enables to access larger volumes of data on more patients and thus provide finer statistics.

For example, patient A has just come back from a tropical country B. He has a rare disease from there. The hospital cannot recognize his disease. The clinic in country B records some cases like his. However, the two databases of the hospital and the clinic are not in the same database engine, or cloud provider. If a cloud federation exists to interconnect the two clouds, his disease could be recognized and he can have a treatment soon.

In cloud federations, pay-as-you-go models and elasticity thus raise an important issue in terms of Multi-Objective Optimization Problems (MOOPs) according to users preferences, such as time, money, quality, etc. However, MOOPs in a cloud federation are hard to solve due to issues of heterogeneity, and variability of the cloud environment, and high complexity in generating the space of candidate solutions.

Let's consider a query **Q** in a example below.

Example 1. A query **Q** in the medical domain, based on TPC-H[3] query 3 and 4:

```
SELECT p.UID, p.PatientID, s.PatientName,
p.PatientBrithDate, p.PatientSex,
p.EthnicGroup, p.SmokingStatus,
s.PatientAge, s.PatientWeight,
s.PatientSize, i.GeneralName,
i.GeneralValues, q.UID,
q.SequenceTags, q.SequenceVRs,
q.SequenceNames, q.SequenceValues
FROM Patient p, GeneralInfoTable i,
Study s, SequenceAttributes q
WHERE p.UID = s.UID AND p.UID = i.UID
AND p.UID = q.UID AND p.PatientSex = 'M'
AND p.SmokingStatus = 'NO' AND s.PatientAge >= x
AND q.SequenceNames
LIKE '%X-ray%'
```

[3] http://www.tpc.org/tpch/

Table 1: Multiple Objectives for Query Execution Plans

QEP	VMs	Price ($/60$min$)	Time (min)	Monetary ($)
QEP1	20	0.02	60	0.4
QEP2	80	0.02	22	0.59
QEP3	50	0.02	26	0.43

This query is transformed into a logical query plan using logical operation, such as *select, project, join, etc*. Depending on the physical operators, a query optimizer generates a query execution plan to execute a logical plan. Actually, various Query Execution Plans (QEPs) are generated with respect to the number of nodes, their capacity in terms of CPU, memory and disk and the pricing model. Table 1 presents an example of possible QEPs for **Q**. Choosing an execution plan is a trade-off between objectives such as the response time or the monetary cost, and depends on users' preferences: a user A may prefer minimizing his budget (QEP1); a user B may want the lowest response time (QEP2); a user C may look for a trade-off between time and money (QEP3).

Assuming that the query is processed on Amazon EC2. The master consists of a m2.4xlarge instance (8 virtual cores with 68.4 GB of RAM). Workers consist of m3.2xlarge instances (8 virtual cores and with 30 GB of RAM). If the pool of resources is 70 VCPU with 260GB of memory, the number of QEPs is thus 70 x 260 = 18,200. The problem is then how to search and optimize such a query in a real environment, when the pool of resources is more variable, with respect to multiple dimensions (response time, monetary cost, etc.). Since generating QEPs maybe infeasible due to high complexity, we aim to find an approximate optimal solution.

In this paper, we address several challenges for the development of medical data management in cloud federations. The first challenge is how to estimate accurate parameter values for MOOPs without precise knowledge of the execution environment in a cloud federation consisting of different sites. The execution environment may consist of various hardware and systems. In addition, it also depends on the variety of physical machines, load evolution and wide-range communications. As a consequence, the estimation process is complex. The second challenge is how to find an approximate optimal solution in MOOPs using an efficient Multi-Objective Optimization algorithm. Indeed, MOOPs could be solved by Multi-Objective Optimization algorithms or the Weighted Sum Model (WSM) [24] or be converted to a Single-Objective Optimization Problem (SOOP). However, SOOPs cannot adequately represent MOOPs [23]. Also, MOOPs leads to find solutions by Pareto dominance techniques. Since generating a Pareto-optimal front is often infeasible due to high complexity [55], MOOPs need an approximate optimal solution calculated by Pareto dominance techniques.

The estimation process can be classified into two classes: without [39, 42, 50] and with machine learning algorithms [17]. In a cloud federation with variability

and different systems, cost functions may be quite complex. In the first class, cost models introduced to build an optimal group of queries [39] are limited to MapReduce [12]. Besides, a PostgreSQL cost model [50] aims to predict query execution time for this specific relational DBMS. Moreover, OptEx [42] provides estimated job completion times for Spark[4] with respect to the size of the input dataset, the number of iterations, the number of nodes composing the underlying cloud. These works mention the estimation of only execution time for a job, and not for other metrics, such as monetary cost. Meanwhile, various machine learning techniques are applied to estimate execution time in recent research [2, 21, 46, 51]. They predict the execution time by many machine learning algorithms. They treat the database system as a black box and try to learn a query running time prediction model using the total information for training and testing in the model building process. It may lead to the use of expired information. In addition, most of these solutions solve the optimization problem with a scalar cost value and do not consider multi-objective problems.

A well known Pareto dominance technique to solve the high complexity of MOOP is Evolutionary Multiobjective Optimization (EMO). Among EMO approaches, Non-dominated Sorting Genetic Algorithms (NSGAs) [14, 15] have lower computational complexity than other EMO approaches [15]. However, these algorithms still have high computational complexity. We presented Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) [36] to improve both computation time and qualities of NSGAs. It has more advantages than other NSGAs. Two versions of NSGA-G will be shown to compromise computation and quality.

In this paper, we introduce a medical system on a cloud federation called Medical Data Management System (MIDAS). It is based on the Intelligent Resource Scheduler (IReS) [17], an open source platform for complex analytics workflows executed over multi-engine environments. In particular, we focus on: (1) a dynamic estimation and (2) a Non-dominated Sorting Genetic Algorithm for Multi-Objective Optimization Problems. The first contribution is Dynamic linear REgression AlgorithM (DREAM) to provide accurate estimation with low computational cost. DREAM is then implemented and validated with experiments on a decision support benchmark (TPC-H benchmark). The second contribution is Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) to improve both quality and computational efficiency of NSGAs, and also provides an alternative for Pareto-optimal of MOOPs. NSGA-Gs are validated through experiments on DTLZ problems [16] and compared with NSGA-II [15], NSGA-III [14], and the others in Generational Distance [49], Inverted Generational Distance [9], and Maximum Pareto Front Error [48] statistic.

This paper is an extended version of [36, 37]. In particular, they are grouped together to become a system, MIDAS. Besides, the theory of NSGA-G in [36] is expanded to two versions: NSGA-G using Min point and using Random metric. The remaining of this paper is organized as follows. Section 2 presents the

[4] `https://spark.apache.org/`

research background. DREAM is presented in Section 3. Section 4 shows NSGA-G. Section 5 presents experiments to validate DREAM and NSGA-Gs. Finally, Section 6 concludes this paper and lists some perspectives.

2 Background

In this section, we introduce an architecture of the system, concepts and techniques, allowing us to implement the proposed medical data management on a cloud federation. First of all, an overview of the Medical Data Management System (**MIDAS**) and the benefits of cloud federation where our system is built on are introduced. After that, an open source platform, which helps our system managing and executing workflows over multi-engine environments is described. The concept of Pareto plan set related to Multi-Objective Optimization Problem (MOOP) in **MIDAS** is then defined. In addition, Multiple Linear Regression and Non-dominated Sorting Genetic Algorithm are also introduced as the basic foundation of our proposed algorithms for MOOP.

2.1 Cloud federation

This section shows the definition and the example related to a cloud federation. A cloud federation enables to interconnect different cloud computing environments. Cloud computing [3] allows to access on demand and configurable resources, which can be quickly made available with minimal maintenance. According to the pay-as-you-go pricing model, customers only pay for resources (storage and computing) that they use. Cloud Service Providers (CSP) supply a pool of resources, development platforms and services. There are many CSPs on the market, such as Amazon, Google and Microsoft, etc., with different services and pricing models. For example, Table 2 shows the pricing of instances in two cloud providers in 2019. The price of Amazon instances are lower than the price of Microsoft instances, but the price of Amazon is without storage. Hence, depending on the demand of a query, the monetary cost is low or high at a specific provider.

In the medical domain, cloud federation may lead to query data across different clouds. A demand query running in that cloud federation could be concerned about the price of time and money of the execution query. It is a Multi-Objective Optimization Problem (MOOP). For example, federating resources makes it possible to access any information on a person with distributed hospital data on various sites. Various big data management system could be used to manage the medical data, which has the 3Vs characteristics of Big Data [1]: high volume, high variety, and high velocity. The data also stores that belong in different clouds are shown in Fig. 1. This example shows that the data can be stored in three different clouds, such as Amazon Web Services, Microsoft Azure, Google Cloud Platform. Pay-as-you-go models in clouds lead to solving Multi-Objective Optimization Problem to find a Query Execution Plan (QEP) according to users preferences, such as time, money, quality, etc. MOOPs often use Pareto dominance techniques in finding an optimal solution.

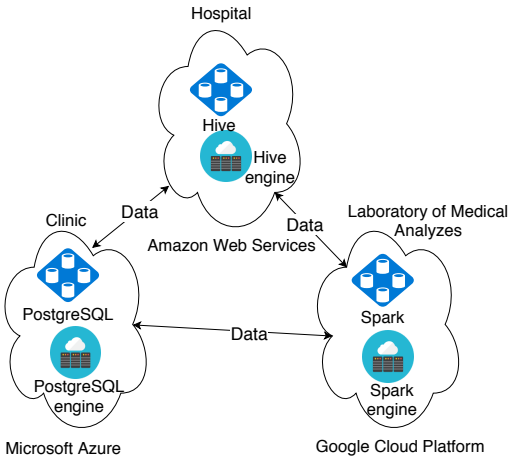

Fig. 1: Motivating Example on using cloud federation.

Table 2: Example of instances pricing in 2019.

Provider	Machine	vCPU	Memory (GiB)	Storage (GB)	Price ($/hour)
Amazon	a1.medium	1	2	EBS-Only	0.0255
	a1.large	2	4	EBS-Only	0.0510
	a1.xlarge	4	8	EBS-Only	0.1020
	a1.2xlarge	8	16	EBS-Only	0.2040
	a1.4xlarge	16	32	EBS-Only	0.4080
Microsoft	B1S	1	1	2	0.0104
	B1MS	1	2	4	0.0208
	B2S	2	4	8	0.0416
	B2MS	2	8	16	0.0832
	B4MS	4	16	32	0.1660
	B8MS	8	32	64	0.3330

2.2 Pareto plan set

Pareto dominance techniques are often used in Multi-Objective Optimization Problem (MOOP), such as Evolutionary Multiobjective Optimization (EMO) [14, 15, 27, 31, 44, 53, 54]. In the vast space of candidate solutions of Multi-Objective Optimization Problem (MOOP), a candidate solution may be not better than another one because of trade-off between various objective values. Pareto sets are used in this situation to optimize a MOOP.

In particular, in a query processing problem, let a **query** q be an information request from databases, presented by a set of tables. A **Query Execution Plan** (QEP), denoted by p, is the evaluation of a query that can be passed to the executor. The set of QEPs is denoted by symbol $\mathcal{P}$. The set of operators is denoted by $\mathcal{O}$. A QEP, p, can be divided into two **sub-plans** p_1 and p_2 if p is the result of function $Combine(p_1, p_2, o)$, where $o \in \mathcal{O}$.

The execution cost of a QEP depends on parameters, which values are not known at the optimization time. A vector $\mathbf{x}$ denotes parameters value and the **parameter space** $\mathcal{X}$ is the set of all possible parameter vectors $\mathbf{x}$. N is denoted as the set of n cost metrics. We can compare QEPs according to n cost metrics which are processed with respect to the parameter vector $\mathbf{x}$ and cost functions $c^n(p, \mathbf{x})$. Let denote $\mathbf{C}$ as the set of cost function c.

Let $p_1, p_2 \in \mathcal{P}$, p_1 **dominates** p_2 if the cost values according to each cost metric of plan p_1 is less than or equal to the corresponding values of plan p_2 in all the space of parameter $\mathcal{X}$. That is to say:

$$\mathbf{C}(p_1, \mathcal{X}) \preceq \mathbf{C}(p_2, \mathcal{X}) \mid \forall n \in N, \forall \mathbf{x} \in \mathcal{X} : c^n(p_1, x) \leq c^n(p_2, x). \tag{1}$$

The function $Dom(p_1, p_2) \subseteq \mathcal{X}$ yields the parameter space region where p_1 dominates p_2 [47]:

$$Dom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) \leq c^n(p_2, x)\}. \tag{2}$$

Assume that in the area $\mathbf{x} \in \mathcal{A}, \mathcal{A} \subseteq \mathcal{X}$, p_1 dominates p_2, $\mathbf{C}(p_1, \mathcal{A}) \preceq \mathbf{C}(p_2, \mathcal{A})$, $Dom(p_1, p_2) = \mathcal{A} \subseteq \mathcal{X}$. p_1 **strictly dominates** p_2 if all values for the cost functions of p_1 are less than the corresponding values for p_2 [47], i.e.

$$StriDom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) < c^n(p_2, x)\}. \tag{3}$$

A **Pareto region** of a plan is a space of parameters where there is no alternative plan has lower cost than it [47]:

$$PaReg(p) = \mathcal{X} \setminus (\bigcup_{p^* \in \mathcal{P}} StriDom(p^*, p)). \tag{4}$$

2.3 IReS

Cloud federation model needs to integrate cloud services from multiple cloud providers. It raises an important issue in terms of heterogeneous database engines in various clouds. Among various heterogeneous database system described in Table 3, IReS platform considers both heterogeneous systems and MOOP in clouds.

Intelligent Multi-Engine Resource Scheduler (IReS) [17] is an open source platform for managing, executing and monitoring complex analytics workflows. IReS provides a method of optimizing cost-based workflows and customizable resource management of diverse execution and various storage engines. Especially, IReS platform helps us to organize data in the multiple clouds as a cloud federation. **Interface** is the first module which is designed to receive information on data and operators, as shown in Fig. 4. The second module is **Modelling**, as shown in Fig. 4, is used to predict the execution time by a model chosen by comparing machine learning algorithms. For example, Least squared regression [41], Bagging predictors [5], Multilayer Perceptron in WEKA framework[5] are used to build the cost model in **Modelling** module. The module

[5] `https://www.cs.waikato.ac.nz/ml/weka/`

Table 3: Recent heterogeneous database system researches.

Research	Heterogeneous	MOOP
Proteus [28]	✓	×
Polystore Query rewriting [40]	✓	×
BigDAWG Polystore System [18]	✓	×
ClooudMdsQL [33, 32]	✓	×
MuSQLE [22]	✓	×
MISO [38]	✓	×
Polybase [11]	✓	×
Estoscada [6]	✓	×
IReS	✓	✓

tests many algorithms and the best model with the smallest error is selected. It guarantees the predicted values as the best one for estimating process. Next module, **Multi-Objective Optimizer**, optimizes Multi-Objective Query Processing (MOQP) and generates a Pareto QEP set. In Multi-Objective problem, the objectives are the cost functions user concerned, such as the execution time, monetary, intermediate data, etc. Multi-Objective Optimization algorithms can be applied to the **Multi-Objective Optimizer**. For instance, the algorithms based on Pareto dominance techniques [10, 14, 15, 27, 31, 36, 44, 53, 54] are solutions for Multi-objective Optimization problems. Finally, the system selects the best QEP based on user query policy and Pareto set. The final QEP is run on multiple engines, as shown in Fig. 4.

2.4 Multiple Linear Regression

In many database management systems, predicting cost values is useful in optimization process [50]. Recent researches have been exploring the statistical machine learning approaches to build predictive models for this task. They often use historical data to train and test the cost model as a Single-Objective Problem (SOP). Besides, Linear Regression is an useful class of models in science and engineering [43]. In this section, we describe the background of this model.

This model is used in the situation in which a cost value, c, is a function of one or more independent variables $x_1, x_2, ...,$ and x_L. For example, execution time c is a function of data size x_1 of first element in join operator and data size x_2 of second element in that join operator.

Given a sample of c values with their associated values of x_i, $i = 1, 2, ..., L$. We focus in the estimation the relationship between c and the independent variables $x_1, x_2, ...,$ and x_L based on this sample. Cost function c of Multiple Linear Regression (MLR) model [43] is defined as follows:

$$c = \beta_0 + \beta_1 x_1 + ... + \beta_L x_L + \epsilon, \tag{5}$$

where β_l, $l = 0, ..., L$, are unknown coefficients, $x_l, l = 1, ..., L$, are the independent variables, e.g., size of data, computer configuration, etc., c is cost function

values and ϵ is random error following normal distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2. The **fitted equation** is defined by:

$$\hat{c} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_L x_L. \tag{6}$$

Example 2. A query **Q** [37] could be expressed as follows:

```
SELECT p.PatientSex, i.GeneralNames
FROM Patient p, GeneralInfo i
WHERE p.UID = i.UID
```

where Patient table is stored in cloud A and uses Hive [45] database engine[6], while GeneralInfo table is in cloud B with PostgreSQL database engine[7]. This scenario leads to concern two metrics of monetary cost and execution time cost. We can use the cost functions which depend on the size of tables of Patient and GeneralInfo. Besides, the configuration and pricing of virtual machines cloud A and B are different. Hence, the cost functions depend on the size of tables and the number of virtual machines in cloud A and B.

$$\hat{c}^{ti} = \hat{\beta}_{t0} + \hat{\beta}_{t1} x_{Pa} + \hat{\beta}_{t2} x_{Ge} + \hat{\beta}_{t3} x_{nodeA} + \hat{\beta}_{t4} x_{nodeB}$$

$$\hat{c}^{mo} = \hat{\beta}_{m0} + \hat{\beta}_{m1} x_{Pa} + \hat{\beta}_{m2} x_{Ge} + \hat{\beta}_{m3} x_{nodeA} + \hat{\beta}_{m4} x_{nodeB}$$

where $\hat{c}^{ti}, \hat{c}^{mo}$ are execution time and monetary cost function; x_{Pa}, x_{Ge} are the size of Patient and GeneralInfo tables, respectively, and x_{nodeA}, x_{nodeB} are the number of virtual machines created to run query **Q**.

There are M historical data, each of them associates with a response c_m, which can be predicted by a **fitted value** $\hat{c}_m$ calculated from corresponding x_{lm} as follows:

$$\hat{c}_m = \hat{\beta}_0 + \hat{\beta}_1 x_{1m} + ... + \hat{\beta}_L x_{Lm}; m = 1, ..., M. \tag{7}$$

Let denote

$$A = \begin{bmatrix} 1 & x_{11} & x_{21} & ... & x_{L1} \\ 1 & x_{12} & x_{22} & ... & x_{L2} \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & x_{1M} & x_{2M} & ... & x_{LM} \end{bmatrix}, \tag{8}$$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_M \end{bmatrix}, \tag{9}$$

$$B = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ . \\ . \\ \hat{\beta}_L \end{bmatrix}. \tag{10}$$

[6] http://hive.apache.org/
[7] https://www.postgresql.org/

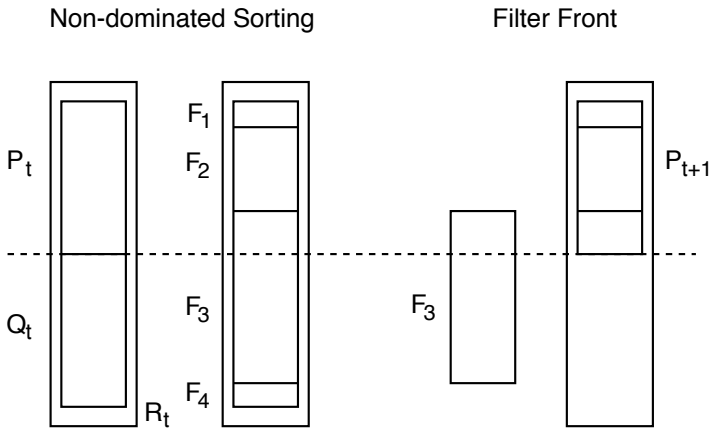

Fig. 2: NSGA-II and NSGA-III procedure [14, 15].

To minimize the **Sum Square Error** (SSE), defined by:

$$SSE = \sum_{m=1}^{M} (c_m - \hat{c}_m)^2, \tag{11}$$

the solution for B is retrieved by:

$$B = (A^T A)^{-1} A^T C. \tag{12}$$

2.5 NSGA

After having the prediction cost values of MOOPs, we need to use Multi-Objective Optimization algorithms to find an optimal solution.

Among Multi-objective Optimization algorithm classes, Evolutionary Multi-objective Optimization (EMO) shows their advantages in searching and optimizing for the MOOPs [10]. Among EMO approaches, Non-dominated Sorting Genetic Algorithms provide low computational complexity of non-dominated sorting, $O(MN^2)$ of NSGAs [14, 15] comparing to $O(MN^3)$ of other Evolutionary Multi-Objective Optimization (EMO), where M is the number of objectives and N is the population size.

NSGA process Initially, NSGAs start with a population P_0 consisting of N solutions. In MOOPs, a population represents a set of candidate solutions. The size of P_0 is smaller than the number of all candidate solutions. Each solution is on a specific rank or non-domination level (any solution in level 1 is not dominated, any solution in level 2 is dominated by one or more solutions in level 1 and so on). At first, the offspring population Q_0 containing N solutions, is created by the binary tournament selection and mutation operators [13]. Where

the binary tournament selection is a method of selecting an individual from a population of individuals in a genetic algorithm, and the mutation operation is a method to choose a neighboring individual in the locality of the current individual. Secondly, a population $R_0 = P_0 \cup Q_0$ with the size of $2N$ will be divided into subpopulations based on the order of Pareto dominance. The appropriate N members from R_0 will be chosen for the next generation. The non-dominated sorting based on usual domination principle [8] is first used, which classifies R_0 into different non-domination levels ($\mathcal{F}_1$, $\mathcal{F}_2$ and so on). After that, a parent population of next-generation P_1 is selected in R_0 from level 1 to level k so that the size of $P_1 = N$ and so on.

The difference among NSGA-II, NSGA-III and other NSGAs is the way to select members in the last level $\mathcal{F}_l$. To keep the diversity, NSGA-II [15] and SPEA-II [54] use crowding distance among solutions in their selection. NSGA-II procedure is not suitable for MOO problems and the crowding distance operator needs to be replaced for better performance [26, 34]. Hence, when the population has a high-density area, higher than others, NSGA-II prefers the solution which is located in a less crowded region.

On the other hand, MOEA/D [53] decomposes a multiple objectives problem into various scalar optimization subproblems. The diversity of solutions depends on the scalar objectives. However, the number of neighborhoods needs to be declared before running the algorithm. In addition, the estimation of good neighborhood is not mentioned. The diversity is considered as the selected solution associated with these different sub-problems. Experimental results in [14] show various versions of MOEA/D approaches which fail to maintain a good distribution of points.

An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-Dominated Sorting Approach [14] (NSGA-III) uses different directions to maintain the diversity of solutions. NSGA-III replaces the crowding distance operator by comparing solutions. Each solution is associated to a reference point [14], which impacts the execution time to built the reference points in each generation. The diversity of NSGA-III is better than the others, but the execution time is very high. For instance, with two objectives and two divisions, three reference points will be created, (0.0,1.0), (1.0,0.0) and (0.5,0.5), as shown in Fig. 3. After selection process, the diversity of population is better than NSGA-II with solutions close to three reference points. However, comparing all solutions to each reference point makes the computation time of NSGA-III very high. In addition, NSGAs often compare all solutions to choose good solutions in $\mathcal{F}_l$. Therefore, when the number of solutions or objectives is significant, the time for calculating and comparing is considerable.

Application In some cases, some objectives are homogeneous. In the reason of the homogeneity between the multi-objectives functions, removing an objective do not affect to the final results of MOO problem. In other cases, the objectives may be contradictory. For example, the monetary is proportional to the execution time in the same virtual machine configuration in a cloud. However, cloud

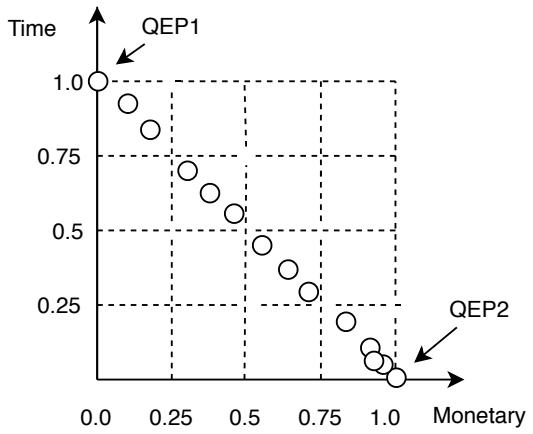

Fig. 3: An example of using the crowing distance in NSGA-II.

providers usually leases computing resources that are typically charged based on a per time quantum pricing scheme [30]. The solutions represent the trade-offs between time and money. Hence, the execution time and the monetary cost cannot be homogeneous.

As a consequence, the multi-objective problem cannot be reduced to a mono-objective problem. Moreover, if we want to reduce the MOO to a mono-objective optimization, we should have a policy to group all objectives by the Weighted Sum Model (WSM) [24]. However, estimating the weights corresponding to different objectives in this model is also a multi-objective problem.

In addition, MOO problems could be solved by MOO algorithms or WSM [24]. However, MOO algorithms are selected thanks to their advantages when comparing with WSM. The optimal solution of WSM could be unacceptable, because of an inappropriate setting of the coefficients [20]. Furthermore, the research in [29] proves that a small change in weights may result in significant changes in the objective vectors and significantly different weights may produce nearly similar objective vectors. Moreover, if WSM changes, a new optimization process will be required.

In conclusion, MOOP approaches leads to using Pareto dominance techniques. A pareto-optimal front is often infeasible [55]. NSGAs show the advantage in searching a Pareto solution for MOOP in less computational complexity than other EMO [15]. However, they should be improved the quality to solve MOOP when the number of objectives is significant.

2.6 Motivation

In the context of medical data management, the background of concepts and techniques related to cloud federations, we introduce a medical system on a cloud federation called Medical Data Management System (MIDAS). It is based

on the Intelligent Resource Scheduler (IReS) [17], an open source platform for complex analytics work-flows executed over multi-engine environments.

MIDAS It is a medical data management system for cloud federation. The proposal aims to provide query processing strategies to integrate existing information systems (with their associated cloud provider and data management system) for clinics and hospitals. Fig. 4 presents an overview of the system. Integrating the system within a cloud federation allows to choose the best strategy for MOQP. MIDAS can be developed based on the platform which can execute over multi-engine environments on clouds. Fig. 4 also shows an example of MIDAS, where three database engines are installed and run in three clouds of different providers.

We choose IReS platform to consider the advantage as shown in Table 3. IReS platform is installed in every machine in MIDAS. The different cloud resource pools allow the system to run in the most appropriate infrastructure environments. The system can optimize workflows between different data sources on different clouds, such as Amazon Web Services[8], Microsoft Azure[9] and Google Cloud Platform[10]. The proposed system is developed based on the Intelligent Resource Scheduler (IReS) for complex analytics workflows executed over multi-engine environments on a cloud federation.

Machine learning algorithm The machine learning algorithms in IReS need entire training datasets to estimate the running costs, which are calculated by determining the cost of processing a job. It may lead to use expired information. Hence, the proposal algorithm aims to improve the accuracy of estimated values with low computational cost. Our proposed method is integrated into IReS to predict the cost values with low computational cost in MOQP of a cloud environment.

Multi-Objective Optimization In addition, MOQP could be solved by Multi-Objective Optimization algorithms or the Weighted Sum Model (WSM) [24]. However, Multi-Objective Optimization algorithms may be selected thanks to their advantages when comparing with WSM. The optimal solution of WSM could be not acceptable, because of an inappropriate setting of the coefficients [20]. Furthermore, the research in [29] proves that a small change in weights may result in significant changes in the objective vectors and significantly different weights may produce nearly similar objective vectors. Moreover, if WSM changes, a new optimization process will be required. Hence, our system applies a Multi-Objective Optimization algorithm to the **Multi-Objective Optimizer** to find a Pareto-optimal solution. When the WSM changes, the final result just is determined by using the Pareto-optimal set at the final step.

[8] `https://aws.amazon.com/`
[9] `https://azure.microsoft.com/`
[10] `https://cloud.google.com/`

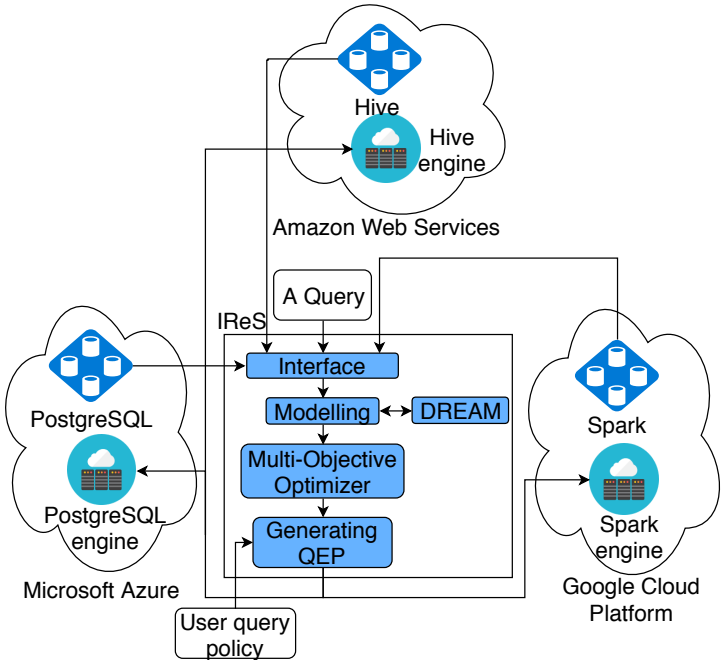

Fig. 4: Architecture of MIDAS [37].

Furthermore, generating a Pareto-optimal front is often infeasible due to high complexity [55]. MOOPs leads to finding an approximate optimal solution by Pareto dominance techniques. A well known approach to solve the high complexity of MOOP is Evolutionary Multiobjective Optimization (EMO). Among EMO approaches, Non-dominated Sorting Genetic Algorithms (NSGAs) [14, 15] have lower computational complexity than other EMO approaches [15]. However, this algorithm still has high computational complexity. We need to find an approach to improve the computational complexity and quality of NSGAs.

In conclusion, our solutions aim to improve the accuracy of cost value prediction with low computational cost and to solve MOQP by Multi-Objective Optimization algorithm in a cloud federation environment. Besides, we also find a method to search and optimize MOOPs by finding an approximate optimal solution in the high complexity of generating a Pareto-optimal front.

3 Dynamic Regression Algorithm

The first technique in MIDAS relates to the estimation of accurate cost values in the variable environment of a cloud federation. Most of cost models [19, 39, 50] depend on the size of data. Hence, our cost functions are functions of the size of data. In particular, cost function and **fitted value** of Multiple Linear

Regression model are previously defined in Section 2.4. The bigger M for sets $\{c_m, x_{lm}\}$ is, the more accurate MLR model usually is. However, the computer is slowing down when M is too big.

Furthermore, the target of Multi-Objective Query Processing is the Multi-Objective Optimization Problem [53], which is defined by:

$$minimize(F(x) = (f_1(x), f_2(x)..., f_K(x))^T), \tag{13}$$

where $x = (x_1, ..., x_L)^T \in \Omega \subseteq \mathbb{R}^L$ is an L-dimensional vector of decision variables, Ω is the decision (variable) space and F is the objective vector function, which contains K real value functions.

In general, it is hard to find a point in Ω that minimizes all the objectives together. Pareto optimality is defined by trade-offs among the objectives. If there is no point $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$, $x^* \in \Omega$, x^* is called Pareto optimal and $F(x^*)$ is called a Pareto optimal vector. Set of all Pareto optimal points is the Pareto set. A Pareto front is a set of all Pareto optimal objective vectors. Generating the Pareto-optimal front can be computationally expensive [55]. In cloud environment, the number of equivalent query execution plans is multiplied.

Example 3. Assuming that a query is processed on Amazon EC2. If the pool of resources includes 70 vCPUs and 260GB of memory, we assume that a configuration to execute a query plan is created by the number of vCPUs and the size of memory which is the multiple of 1GB. In particular, a configuration can be 01 vCPU and 260GB of memory and the others is 70 vCPUs and 01GB of memory. Hence, the combination of different configurations to execute this query would be 70*260=18,200.

Example 3 shows that a query plan can generate multiple equivalent QEPs in cloud environment. The smaller M for sets $\{c_m, x_{lm}\}$ is, the faster speed for the estimation cost process of Multi-Objective Query Processing for a QEP is. In the system of computationally expensiveness in cloud environment as in Example 3, a small reduction of computation for an equivalent QEP estimation will become significant for a large number of equivalent QEPs estimation.

The most important idea is to estimate MLR quality by using the coefficient of determination. The coefficient of determination [43] is defined by:

$$R^2 = 1 - SSE/SST, \tag{14}$$

where SSE is the sum of squared errors and SST represents the amount of total variation corresponding to the predictor variable X. Hence, R^2 shows the proportion of variation in cost given by the Multiple Linear Regression model of variable X. For example, the model gives $R^2 = 0.75$ of time response cost, it can be concluded that 3/4 of the variation in time response values can be explained by the linear relationship between the input variables and time response cost. Table 4 presents an example of MLR with different number of measures. The smallest dataset is $M = L + 2 = 4$ [43], where M is the size of previous data and L is the number of variables in Equation (5). In general, R^2 increases in parallel

Table 4: Using MLR in different size of dataset [37].

Cost	x_1	x_2	M	R^2
20.640	0.4916	0.2977		
15.557	0.6313	0.0482		
20.971	0.9481	0.8232		
24.878	0.4855	2.7056	4	0.7571
23.274	0.0125	2.7268	5	0.7705
30.216	0.9029	2.6456	6	0.8371
29.978	0.7233	3.0640	7	0.8788
31.702	0.8749	4.2847	8	0.8876
20.860	0.3354	2.1082	9	0.8751
32.836	0.8521	4.8217	10	0.8945

with M. In particular, R^2 should be greater than 0.8 to provide a sufficient quality of service level. As a consequence, M should be greater than 5 to provide enough accuracy. Hence, when the system requires the minimum values of R^2 is equal to 0.8, $M > 6$ is not recommended. In general, R^2 still rises up when M goes up. Therefore, we need to determine the model which is sufficient suitable by the coefficient of determination.

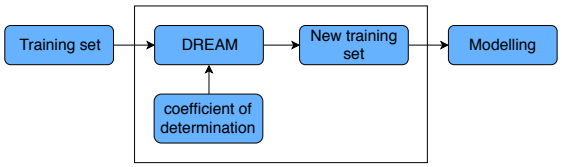

Fig. 5: DREAM module [37].

Our motivation is to provide accurate estimation while reducing the number of previous measures based on R^2. We thus propose DREAM as a solution for cloud federation and their inherent variance, as shown in Fig. 5. DREAM uses the training set to test the size of new training dataset. It depends on the predefined coefficient of determination. The new training set is generated in order to have the updated value and avoid using the expired information. With the new training set, **Modelling** uses fewer data in the building model process than the original approach does.

Cost modeling without machine learning [39, 42, 50] often uses the size of data to estimate the execution time for the specific system. Besides, the machine learning approach [17] can use any information to estimate the cost value. Hence, our algorithm uses the size of data as variables of DREAM. In (6), $\hat{c}$ is the cost value, which needs to be estimated in MOQP, and $x_1, x_2, \ldots$ are the information of system, such as size of input data, the number of nodes, the type of virtual machines. If $R^2 \geq R^2_{require}$, where $R^2_{requires}$ is predefined by users, the

Algorithm 1 Calculate the predict value of multi-cost function [37]

function EstimateCostValue($\mathbf{R}^2_{require}, X, M_{max}$)
  for $n = 1$ to N **do**
    $\mathbf{R^2_n} \leftarrow \emptyset$ //with all cost function
  end for
  $m = L + 2$ //at least $m = L + 2$
  while (any $R^2_n < R^2_{n-require}$) and $m < M_{max}$ **do**
    for $\hat{c}_n(p) \subseteq \mathbf{\hat{c}_N(p)}$ **do**
      $R^2_n = 1 - SSE/SST$
      $\hat{c}_n = \hat{\beta}_{n0} + \hat{\beta}_{n1}x_1 + ... + \hat{\beta}_{nL}x_L$
    end for
    $m = m + 1$
  end while
  return $\mathbf{\hat{c}_N(p)}$
end function

model is reliable. In contrast, it is necessary to increase the number of set value. Algorithm 1 shows a scheme as an example of increasing value set: $m = m + 1$.

In this paper, we focus on the accuracy of execution time estimation with the low computational cost in MOQP. The original optimization approach in IReS uses Weighted Sum Model (WSM) [24] with user policy to find the best candidate solution. However, the optimal solution of WSM could be not acceptable, because of an inappropriate setting of the coefficients [20]. Besides, Multi-Objective Optimization algorithms have more advantages than WSM [20, 29]. They lead to find solutions by Pareto dominance techniques. However, generating a Pareto-optimal front is often infeasible due to high complexity [55]. One of well known Multi-Objective Optimization algorithm class is Non-dominated Sorting Algorithms (NSGAs). Hence, after having a set of predicted cost function values for each query plan, a Multi-Objective Optimization algorithm, such as NSGA-G [36] is applied to determine a Pareto query execution plan set. At the final step, the weight sum model **S** and the constraint **B** associated with the user policy are used to return the best QEP for the given query [24]. In particular, the most meaningful plan will be selected by comparing function values with weight parameters between $\hat{c}_n$ [24] at the final step, as shown in Algorithm 2. Fig. 6 shows the different between two MOQP approaches. Our algorithms are developed based on the MLR described above using x_i for size of data and c_i for the metric cost, such as the execution time, energy consumption, etc.

4 Non-dominated Sorting Genetic Algorithm based on Grid partitioning

After having the prediction cost values of MOOPs by DREAM, we need to use Multi-Objective Optimization algorithms to find an optimal solution. Hence, the second technique relates to looking for an efficient approach for searching and optimizing in MIDAS is introduced in this section. NSGAs [14, 15] are well

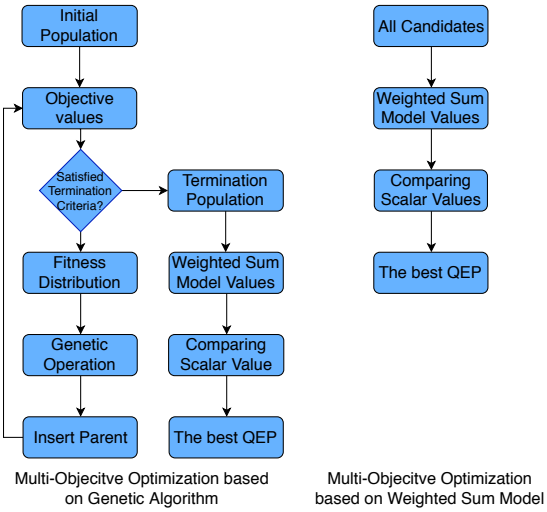

Fig. 6: Comparing two MOQP approaches [37]

Algorithm 2 Select the best query plan in $\mathcal{P}$ [37]

1: **function** BESTINPARETO($\mathcal{P}, \mathbf{S}, \mathbf{B}$)
2: $P_B \leftarrow p \in \mathcal{P} | \forall n \leq |\mathbf{B}| : c_n(p) \leq B_n$
3: **if** $P_B \neq \emptyset$ **then**
4: **return** $p \in P_B | C(p) = min(WeightSum(P_B, \mathbf{S}))$
5: **else**
6: **return** $p \in \mathcal{P} | C(p) = min(WeightSum(\mathcal{P}, \mathbf{S}))$
7: **end if**
8: **end function**

known approaches to optimize MOOPs. Our previous work [36] proposed Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) to improve both diversity and convergence of NSGAs while having an efficient computation time by reducing the space of selected good solutions in the truncating process. NSGA-G is an algorithm based on genetic algorithms (GAs). The convergence of GAs is discussed in [7]. The difference between many GAs is the qualities of diversity and convergence. We will describe the strategy to improve the qualities of NSGAs while having an efficient computation time as below.

At the t^{th} generation of Non-dominated Sorting Genetic Algorithms, P_t represents the parent population with N size and Q_t is offspring population with N members created by P_t. $R_t = P_t \cup Q_t$ is a group in which N members will be selected for P_{t+1}.

4.1 Main process

This section describes more details about the main process of NSGAs. Algorithm 3 shows the steps of the processing. First, the Offspring is initialized in

Algorithm 3 Main process [14, 15].

```
1: function ITERATE(Population)
2:     Offsprings ← ∅
3:     while Offsprings.size < populationSize do
4:         Parent = Selection(Population)
5:         Offsprings = Offsprings ∪ Evolve(Parent)
6:     end while
7:     Population = Population ∪ Offsprings
8:     Population = Truncate(Population)
9:     return Population
10: end function
```

Algorithm 4 Non-dominated Sorting [14].

```
Require: R
1: function SORTING(R)
2:     RinRank ← ∅
3:     rank = 1
4:     remaining ← R
5:     while RisNotEmpty do
6:         Front ← non − dominatedPopulation(remaining, rank)
7:         remaining = remaining \ Front
8:         RinRank = RinRank ∪ Front
9:         rank + +
10:    end while
11:    return RinRank
12: end function
```

Line 2. The size of Offspring equals to the size of Population, i.e., N. Hence, a parent is selected from the population and evolved to become a new offspring. A new population with the size of $2N$ is created from Offspring and the old population. After that, the function Truncate will cut off the new population to reduce the members to the size of N, as shown in Line 8.

4.2 Non-Dominated Sorting

Before the truncating process, the solutions in the population with a size of $2N$ should be sorted in multiple fronts with their ranking, as shown in Algorithm 4. First, the Non-dominated sorting operator generates the first Pareto set in a population of $2N$ solutions. Its rank is 1. After that, the process is repeated until the remain population is empty. Finally, $2N$ solutions are divided into various fronts with their ranks.

4.3 Filter front process

NSGA-G using Min point NSGA-G finds the nearest smaller and bigger grid point for each solution. For example, Fig. 7 shows an example of a two-objective

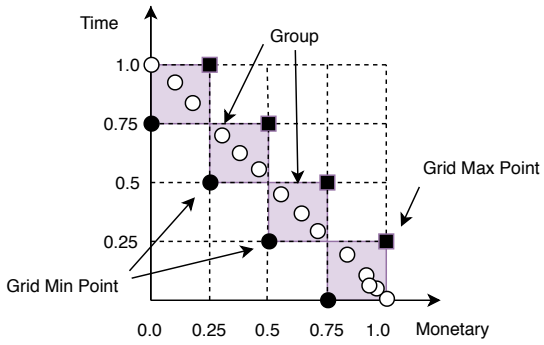

Fig. 7: An example of using Grid points.

Algorithm 5 Filter front in NSGA-G using Min point. [36]

1: **function** FILTER($\mathcal{F}_l, M = N - \sum_{j=1}^{l-1} \mathcal{F}_j$)
2: updateIdealPoint()
3: updateIdealMaxPoint()
4: translateByIdealPoint()
5: normalizeByMinMax()
6: createGroups
7: **while** $\mid \mathcal{F}_l \mid > M$ **do**
8: selectRandomGroup()
9: removeMaxSolutionInGroup()
10: **end while**
11: **return** $\mathcal{F}_l$
12: **end function**

problem. If the unit of the grid point is 0.25 (the size of grid is 4) and the solution with two-objective value is $[0.35, 0.65]$, the closest Grid Min Point is $[0.25, 0.5]$ and the nearest Grid Max Point is $[0.5, 0.75]$.

The first strategy avoids computing multiple objective cost values of all solutions in the population, the space is divided into multiple small groups by Grid Min Point and Grid Max Point, as shown in Fig. 7. Each group has one Grid Min Point, the nearest smaller point and one Grid Max Point, the nearest bigger point. Only solutions in a group are calculated and compared. The solution has the smallest distance to the nearest smaller point in a group will be added to P_{t+1}. In this way, in any loop, we do not need to calculate the crowding-distance values or estimate the smallest distance from solutions to the reference points among all members in the last front, as shown in Fig. 7. In any loop, it is not necessary to compare solutions among all members in F_l, as F_3 in Fig. 2. The second strategy chooses randomly a group. The characteristic of diversity is maintained by this strategy. Both strategies are proposed to improve the qualities of our algorithm. Algorithm 5 shows the strategy to select $N - \sum_{j=1}^{l-1} \mathcal{F}_j$ members in $\mathcal{F}_l$.

Algorithm 6 Filter front in NSGA-G using Random metric.

```
1: function FILTER(F_l, M = N − Σ_{j=1}^{l−1} F_j)
2:     updateIdealPoint()
3:     updateIdealMaxPoint()
4:     translateByIdealPoint()
5:     normalizeByMinMax()
6:     createGroups
7:     while | F_l |> M do
8:         selectRandomGroup()
9:         selectRandomMetric()
10:        removeWorstSolutionInGroup()
11:    end while
12:    return F_l
13: end function
```

The two lines 2 and 3 in Algorithm 5 determine the new origin coordinates and the maximum objective values of all solutions, respectively. After that, they will be normalized in a range of $[0, 1]$. All solutions will be in different groups, depending on the coefficient of the grid. The most important characteristic of this algorithm is randomly selecting the group like NSGA-III to keep the diversity characteristic and remove the solution among members of that group. This selection helps to avoid comparing and calculating the maximum objectives in all solutions.

To estimate the quality of the proposed algorithm, three qualities, Generational Distance [49], Inverted Generational Distance [9] and the Maximum Pareto Front Error [48], are used.

NSGA-G using Random metric In MOOP, when the number of objectives is significant, any function used to compare solutions leads to high computation. NSGA-G using Min point uses Grid partition to reduce the number in groups, but it still needs a function to group all objectives value to a scalar value. In order to decrease the execution time, this section proposes a random method to compare solutions among a group. This approach does not generate any reference point or an intermediate function to estimate the value of solutions. The natural metric values are chosen randomly to remove the worst solution in the different groups.

All the steps in this algorithm are similar to NSGA-G using Min point, as shown from Line 2 to Line 6. Loop *While* has one more step of choosing metric randomly. Function *selectRandomMetric* is used to select a natural metric among the objectives in MOOP. The important characteristics of this algorithm are randomly selecting the group like NSGA-G and using natural metric among various objectives. It aims to keep the diversity characteristic, and reduce the comparing time. This selection helps to avoid using an intermediate function in comparing and calculating the values of solutions.

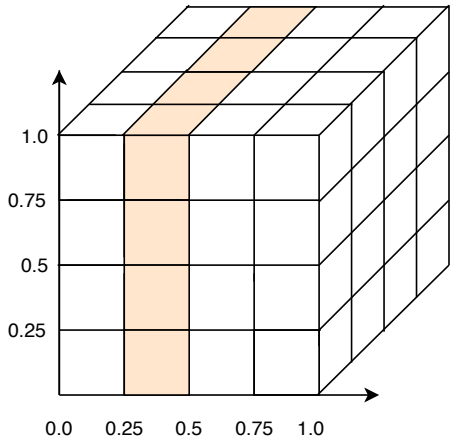

Fig. 8: A simple front group.

4.4 Selecting the size of grid

The proposed approach uses Grid partitioning to guarantee that the solutions are distributed in all the solution space. Assuming that there is a problem with N objectives. The last front should remove k solutions. By normalizing the space of solution in the range of $[0, 1]$ and dividing that range to n segments, a solution belongs to one of n^N groups in that space. In terms of Non-dominated principle, a group including a solution in that space have many other groups which contain Non-dominated solutions. These groups are called *Non-dominated groups*. All the groups in this situation make a set groups, called *front group*.

The proposed idea is to keep the diversity characteristic of the genetic algorithm by generating k groups and removing k solutions. Hence, the ideal *front group* is designed so that it has k groups.

Simple front group From a group in the normalizing space in range of $[0, 1]$, a simple plane covers it and includes *Non-dominated groups*. In the space of N axes, the number of groups is n^N. Hence, the simple *front group* is the simple plane. The number of groups in that *front group* is n^{N-1}. Therefore, if the last front needs to remove k solutions, the number of grid n is determined as follows

$$n = \lceil k^{\frac{1}{N-1}} \rceil. \tag{15}$$

For example, Fig. 8 shows a problem with 3 objectives. In each axis coordinate, the size of grid is 4, and the maximum number of groups in all space of N axis coordinates is 4^3. A simple *front group* includes $4^{3-1} = 16$ groups. If the last front needs to remove 15 solutions, the number of grid when we choose simple front group is $n = \lceil k^{\frac{1}{N-1}} \rceil = 4$.

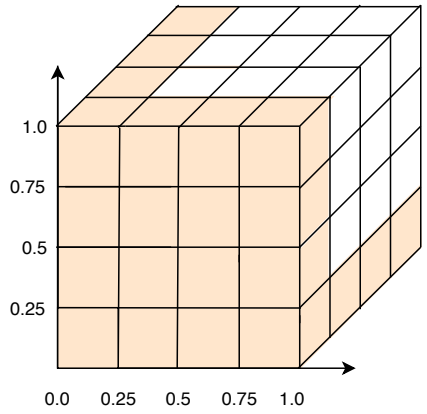

Fig. 9: A max front group.

Max front group From a group in the normalizing space in range of $[0, 1]$, a simple plane covers it and includes *Non-dominated groups*. In the space of N axis coordinates, the number of groups is n^N. *Max front group* has the largest number of groups includes N planes. Hence, the number of groups in *Max front group* is $n^N - (n-1)^N$. Therefore, if the last front needs to remove k solution, the number of grid n is determined as follows

$$n^N - (n-1)^N = k. \tag{16}$$

For instance, Fig. 9 shows a problems with 3 objectives. In each axis coordinate, the size of grid is 4, the maximum number of groups in all space of N axis coordinates is 4^3. *Max front group* includes $4^3 - 3^3 = 64 - 27 = 37$ groups.

5 Validation

5.1 DREAM

The previous section introduces two algorithms for the Multi-Objective Optimization Problem in MIDAS. DREAM and NSGA-G have been implemented on top of IReS platform. They have been validated with experiments.

Implementation Our experiments are executed on Galactica private cloud [11] with a cluster of three machines. Each node has four 2.4 GHz CPU, 80 GiB Disk, 8 GiB memory and runs 64-bit platform Linux Ubuntu 16.04.2 LTS. The system uses Hadoop 2.7.3[12], Hive 2.1.1, PostgreSQL 9.5.14, Spark 2.2.0 and Java OpenJDK Runtime Environment 1.8.0. IReS platform is used to manage data in multiple database engine and deploy the algorithms.

[11] `https://horizon.isima.fr/`
[12] `http://hadoop.apache.org/`

Table 5: Comparison of mean relative error with 100MiB TPC-H dataset [37].

Query	BML_N	BML_{2N}	BML_{3N}	BML	DREAM
12	0.265	0.459	0.220	0.485	0.146
13	0.434	0.517	0.381	0.358	0.258
14	0.373	0.340	0.335	0.358	0.319
17	0.404	0.396	0.267	0.965	0.119

Table 6: Comparison of mean relative error with 1GiB TPC-H dataset [37].

Query	BML_N	BML_{2N}	BML_{3N}	BML	DREAM
12	0.349	0.854	0.341	0.480	0.335
13	0.396	0.843	0.457	0.487	0.349
14	0.468	0.664	0.539	0.790	0.318
17	0.620	0.611	0.681	0.970	0.536

Experiments TPC-H benchmark with two datasets of 100MB and 1GB is used to have experiments with DREAM. Experiments with TPC-H benchmark are executed in a multi-engine environment consisting of Hive and PostgreSQL deployed on Galactica private cloud. In TPC-H benchmark, the queries related to two tables are 12, 13, 14 and 17. These queries with two tables in two different databases, such as Hive and PostgreSQL, are studied.

Results To estimate the quality of price models which are estimated by DREAM in comparison with other algorithms, Mean Relative Error (MRE), a metric used in [2] is used and described as below:

$$\frac{1}{M}\sum_{M}^{i=1}\frac{|\hat{c}_i - c_i|}{c_i}, \tag{17}$$

where M is the number of testing queries, $\hat{c}_i$ and c_i are the predict and actual execution time of testing queries, respectively. IReS platform uses multiple machine learning algorithms in their model, such as Least squared regression, Bagging predictors, Multilayer Perceptron.

In IReS model building process, IReS tests many algorithms and the best model with the smallest error is selected. It guarantees the predicted values as the best one for estimating process. DREAM is compared to the Best Machine Learning model (BML) in IReS platform with many observation window (N, $2N$, $3N$ and no limit of history data). The smallest size of a window, $N = L + 2$ [43], where L is the number of variables, is the minimum data set DREAM requires.

As shown in Table 5 and 6, MRE of DREAM are the smallest values between various observation windows. In our experiments, the size of historical data, which DREAM uses, are small, around N.

5.2 NSGA-G

Various earlier studies on Multiple Objective Evolutionary Algorithms (MOEAs) introduce test problems which are either simple or not scalable. DTLZ test problems [16] are useful in various research activities on MOEAs, e.g., testing the performance of a new MOEA, comparing different MOEAs and a better understanding of MOEAs. The proposed algorithm is experimented on DTLZ test problems with other famous NSGAs to show advantages in convergence, diversity and execution time.

Implementation Our experiments use Multiobjective Evolutionary Algorithms (MOEA)[13] framework in Open JDK Java 1.8. All experiments are run on a machine with following parameters: Intel(R) core(TM) i7-6600U CPU @ 2.60GHz $\times$ 4, 16GB RAM.

Experiments For fair comparison and evaluation, the same parameters are used, such as Simulated binary crossover [13] (30), Polynomial mutation [13] (20), max evaluations (10000) and populations (100) for eMOEA [10], NSGA-II, MOEA/D [53], NSGA-III and NSGA-G[14], during 50 independent running to solve two types of problems in DTLZ test problems [16] with m objectives, $m \in [5, 10]$. These algorithms use the same population size $N = 100$ and the maximum evaluation $M = 10000$. We apply *Simple front group* approach, equation 15, to determine the grid in both of NSGA-Gs with Min point and Random metric experiments. We use the Generational Distance (GD) [49], Inverted Generational Distance (IGD) [9] and the Maximum Pareto Front Error (MPFE) [48] to compare the quality of NSGA-Gs to other NSGAs.

GD measures how far the evolved solution set is from the true Pareto front [52], as shown in following:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \tag{18}$$

where $d_j = \min_j ||f(x_i) - PF_{true}(x_j)||$ shows the distance objective space between solution x_i and the nearest member in the true Pareto front (PF_{true}), and n is the number of solutions in the approximation front. Lower value of GD represents a better quality of an algorithm.

IGD is a metric to estimate the approximation quality of the Pareto front obtained by MOO algorithms [4], which can measure both convergence and diversity in a sense. IGD is shown in the following equation [52]:

$$IGD = \frac{\sum_{v \in PF_{true}} d(v, X)}{|PF_{true}|}, \tag{19}$$

where X is the set of non-dominated solutions in the approximation front, $d(v, X)$ presents the minimum Euclidean distance between a point v in PF_{true}

[13] `http://moeaframework.org/`
[14] `https://gitlab.inria.fr/trle/moea`

and the points in X. Lower value of IDG represents the approximate front getting close to PF_{true}, and not missing any part of the whole PF_{true}.

MPFE shows the most significant distance between the individuals in Pareto front and the solutions in the approximation front [52]. This metric is shown in the following equation:

$$MPFE = \max_i d_i. \tag{20}$$

In all tables show the experiments, the darkest mark value show the least value in various algorithm experiments, and the brighter mark value is the second least value among them.

Study on test problems In this section, we use DTLZs, and WFG [25] test problem to experiment NSGA-Gs. Advantages of two versions of NSGA-G are present in Table 7, 8, 9, 10, 11, and 12. Metrics, such as GD, IDG, MPFE, are used to estimate the qualities of the different algorithms. These experiments compare both of NSGA-Gs with Min point and Random metric to other algorithms.

Table 7: Generational Distance

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	2.595e-01	4.418e-01	2.251e+01	4.264e-01	3.090e+00	1.977e-01
DTLZ3	5	1.861e-01	5.528e-02	1.130e+00	8.650e-02	3.079e-01	1.678e-02
WFG1	5	1.133e-03	9.748e-04	6.923e-03	6.908e-03	3.218e-03	7.617e-04
WFG3	5	4.027e-04	0.000e+00	2.549e-03	1.941e-03	2.011e-03	1.061e-05
DTLZ1	6	2.903e+00	2.137e+00	9.131e+01	1.820e+00	6.839e+00	4.907e-01
DTLZ3	6	2.226e+01	1.332e+01	1.252e+02	1.760e+01	2.389e+01	5.457e+00
WFG1	6	1.207e-03	8.842e-04	8.000e-03	6.753e-03	3.559e-03	7.417e-04
WFG3	6	4.104e-04	0.000e+00	2.523e-03	1.639e-03	1.800e-03	5.384e-05
DTLZ1	7	7.790e-01	8.949e-01	2.228e+01	2.601e-01	1.407e+00	8.201e-02
DTLZ3	7	1.719e-01	4.449e-02	1.309e+00	3.610e-02	1.619e-01	5.628e-03
WFG1	7	1.048e-03	8.219e-04	6.825e-03	5.613e-03	3.891e-03	6.405e-04
WFG3	7	4.011e-04	3.055e-06	2.390e-03	1.871e-03	1.665e-03	5.926e-05
DTLZ1	8	5.823e+00	5.851e+00	1.130e+02	1.276e+00	9.933e+00	4.660e-01
DTLZ3	8	2.071e+01	1.941e+01	1.604e+02	1.355e+01	3.001e+01	4.757e+00
WFG1	8	1.377e-03	9.406e-04	9.023e-03	7.659e-03	4.454e-03	6.469e-04
WFG3	8	3.655e-04	2.689e-05	1.692e-03	1.301e-03	9.662e-04	6.578e-05
DTLZ1	9	8.374e-01	3.626e+00	3.074e+01	3.544e-01	2.772e+00	1.003e-01
DTLZ3	9	4.673e-02	7.112e-02	6.293e-01	8.922e-03	1.052e-01	2.843e-03
WFG1	9	1.309e-03	8.924e-04	8.882e-03	7.551e-03	4.020e-03	6.816e-04
WFG3	9	3.597e-04	2.576e-05	1.298e-03	1.208e-03	7.634e-04	5.365e-05
DTLZ1	10	7.375e-01	1.519e+00	2.091e+01	2.705e-01	2.207e+00	3.021e-02
DTLZ3	10	4.785e-02	1.116e-01	6.793e-01	7.345e-03	1.118e-01	2.939e-03
WFG1	10	1.369e-03	1.385e-03	8.551e-03	6.364e-03	3.648e-03	6.692e-04
WFG3	10	3.259e-04	0.000e+00	1.196e-03	1.265e-03	6.945e-04	4.352e-05

First, two versions of NSGA-G often show that they are faster than the other algorithms in all experiments of average computation time, Table 8, 10, and 12.

Second, NSGA-Gs are also better than other NSGAs in terms of quality in GD and MPFE experiments, as shown in Table 7, and 11. Except for the IDG experiment, as shown in Table 9 the quality of NSGA-G with Random metric is not as good as other ones. However, the fastest algorithm among NSGAs is

Table 8: Average compute time in Generational Distance experiment

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	3.604e+01	6.642e+01	5.508e+01	2.000e+02	2.241e+02	6.366e+01
DTLZ3	5	5.398e+01	6.440e+01	7.074e+01	1.870e+02	2.714e+02	6.212e+01
WFG1	5	1.379e+02	6.658e+01	6.636e+01	1.899e+02	2.594e+02	6.720e+01
WFG3	5	8.562e+02	8.162e+01	6.074e+01	1.864e+02	3.077e+02	8.370e+01
DTLZ1	6	4.552e+01	5.582e+01	5.632e+01	1.918e+02	1.662e+02	5.672e+01
DTLZ3	6	9.340e+01	6.572e+01	6.362e+01	1.971e+02	1.783e+02	6.638e+01
WFG1	6	1.961e+02	9.826e+01	7.392e+01	2.049e+02	2.157e+02	7.286e+01
WFG3	6	1.083e+03	7.580e+01	6.642e+01	1.967e+02	2.384e+02	7.782e+01
DTLZ1	7	6.206e+01	5.834e+01	6.208e+01	2.290e+02	1.621e+02	5.964e+01
DTLZ3	7	1.568e+02	6.992e+01	7.024e+01	2.405e+02	1.817e+02	7.022e+01
WFG1	7	2.585e+02	7.806e+01	8.042e+01	2.473e+02	2.085e+02	7.810e+01
WFG3	7	1.469e+03	8.030e+01	9.184e+01	2.896e+02	2.821e+02	9.950e+01
DTLZ1	8	8.762e+01	5.998e+01	6.640e+01	2.450e+02	2.327e+02	6.244e+01
DTLZ3	8	2.235e+02	7.618e+01	7.652e+01	2.536e+02	2.535e+02	7.424e+01
WFG1	8	3.100e+02	8.034e+01	8.710e+01	2.625e+02	2.924e+02	8.206e+01
WFG3	8	1.464e+03	7.912e+01	7.772e+01	2.542e+02	3.268e+02	8.346e+01
DTLZ1	9	1.157e+02	6.264e+01	7.034e+01	2.524e+02	3.095e+02	6.590e+01
DTLZ3	9	2.978e+02	7.694e+01	8.422e+01	2.678e+02	3.422e+02	7.828e+01
WFG1	9	3.846e+02	8.442e+01	9.426e+01	2.731e+02	3.844e+02	8.668e+01
WFG3	9	1.677e+03	8.954e+01	8.166e+01	2.595e+02	4.373e+02	8.642e+01
DTLZ1	10	1.527e+02	6.510e+01	7.584e+01	2.740e+02	4.204e+02	6.874e+01
DTLZ3	10	3.860e+02	8.132e+01	8.916e+01	2.883e+02	4.641e+02	8.370e+01
WFG1	10	4.747e+02	8.996e+01	1.005e+02	2.941e+02	5.175e+02	9.272e+01
WFG3	10	1.881e+03	8.576e+01	8.640e+01	2.802e+02	6.035e+02	9.128e+01

Table 9: Inverted Generational Distance

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	3.437e-01	1.027e+00	3.741e+01	6.226e-01	3.465e+00	4.637e-01
DTLZ3	5	5.568e-01	4.794e-01	3.576e+00	3.969e-01	1.098e+00	1.589e-01
WFG1	5	1.298e-01	2.924e-01	1.234e-01	7.202e-02	1.365e-01	2.906e-01
WFG3	5	4.167e-02	3.850e-01	1.272e-01	1.417e-01	7.899e-02	3.987e-01
DTLZ1	6	4.975e+00	6.617e+00	2.469e+02	2.903e+00	9.524e+00	2.688e+00
DTLZ3	6	1.131e+02	4.698e+01	5.199e+02	4.207e+01	8.253e+01	2.761e+01
WFG1	6	1.722e-01	3.705e-01	1.531e-01	7.460e-02	1.596e-01	3.341e-01
WFG3	6	5.367e-02	5.424e-01	1.488e-01	1.630e-01	1.065e-01	5.146e-01
DTLZ1	7	7.034e-01	4.042e+00	1.938e+01	4.718e-01	7.695e-01	8.458e-01
DTLZ3	7	7.320e-01	4.310e-01	4.852e+00	2.878e-01	3.826e-01	2.524e-01
WFG1	7	1.437e-01	3.547e-01	1.371e-01	7.114e-02	1.403e-01	3.199e-01
WFG3	7	6.134e-02	6.325e-01	1.573e-01	1.705e-01	1.169e-01	6.122e-01
DTLZ1	8	1.234e+01	1.212e+01	4.166e+02	3.101e+00	1.073e+01	2.849e+00
DTLZ3	8	1.501e+02	6.557e+01	7.623e+02	3.720e+01	1.011e+02	2.665e+01
WFG1	8	1.284e-01	3.186e-01	1.251e-01	6.956e-02	1.238e-01	2.692e-01
WFG3	8	6.487e-02	6.477e-01	1.593e-01	1.704e-01	1.115e-01	6.094e-01
DTLZ1	9	4.009e-01	3.676e+00	5.490e+00	3.932e-01	6.185e-01	5.747e-01
DTLZ3	9	3.029e-01	4.578e-01	1.713e+00	2.398e-01	2.584e-01	2.401e-01
WFG1	9	1.167e-01	2.921e-01	1.193e-01	6.477e-02	1.131e-01	2.561e-01
WFG3	9	6.758e-02	6.897e-01	1.621e-01	1.675e-01	1.078e-01	6.237e-01
DTLZ1	10	9.350e-01	9.074e+00	1.357e+01	6.061e-01	1.499e+00	1.028e+00
DTLZ3	10	4.368e-01	5.440e-01	2.368e+00	2.000e-01	3.965e-01	1.912e-01
WFG1	10	1.147e-01	3.043e-01	1.167e-01	6.273e-02	1.102e-01	2.671e-01
WFG3	10	6.759e-02	6.676e-01	1.670e-01	1.696e-01	1.043e-01	6.102e-01

Table 10: Average compute time in Inverted Generational Distance experiment

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	3.384e+01	5.500e+01	5.176e+01	1.840e+02	2.139e+02	5.276e+01
DTLZ3	5	9.490e+01	8.072e+01	5.954e+01	2.942e+02	2.803e+02	6.146e+01
WFG1	5	1.453e+02	7.752e+01	8.710e+01	1.957e+02	2.988e+02	8.220e+01
WFG3	5	9.067e+02	8.638e+01	5.950e+01	2.087e+02	3.137e+02	8.416e+01
DTLZ1	6	4.982e+01	6.264e+01	5.860e+01	2.209e+02	1.894e+02	6.534e+01
DTLZ3	6	9.604e+01	6.984e+01	6.554e+01	2.182e+02	1.958e+02	7.078e+01
WFG1	6	2.188e+02	8.088e+01	7.810e+01	2.452e+02	2.282e+02	8.362e+01
WFG3	6	2.601e+03	9.036e+01	6.638e+01	3.200e+02	3.094e+02	1.215e+02
DTLZ1	7	6.754e+01	5.880e+01	6.122e+01	2.517e+02	1.620e+02	6.066e+01
DTLZ3	7	1.587e+02	7.172e+01	6.986e+01	2.525e+02	1.798e+02	7.168e+01
WFG1	7	2.579e+02	7.696e+01	8.294e+01	2.587e+02	2.185e+02	7.768e+01
WFG3	7	1.272e+03	7.836e+01	7.194e+01	2.487e+02	2.284e+02	8.888e+01
DTLZ1	8	8.430e+01	5.996e+01	6.610e+01	2.537e+02	2.322e+02	6.328e+01
DTLZ3	8	2.358e+02	7.418e+01	7.808e+01	2.608e+02	2.535e+02	7.446e+01
WFG1	8	3.158e+02	7.960e+01	8.682e+01	2.704e+02	2.903e+02	8.344e+01
WFG3	8	1.432e+03	8.044e+01	7.712e+01	2.513e+02	3.242e+02	8.364e+01
DTLZ1	9	1.237e+02	6.278e+01	6.978e+01	2.563e+02	3.120e+02	6.646e+01
DTLZ3	9	3.174e+02	7.882e+01	8.330e+01	2.721e+02	3.418e+02	7.838e+01
WFG1	9	3.827e+02	8.586e+01	9.338e+01	2.718e+02	3.837e+02	8.594e+01
WFG3	9	1.696e+03	8.290e+01	8.142e+01	2.607e+02	4.369e+02	8.654e+01
DTLZ1	10	1.436e+02	6.472e+01	7.536e+01	2.753e+02	4.187e+02	6.876e+01
DTLZ3	10	4.003e+02	8.566e+01	8.872e+01	2.897e+02	4.572e+02	8.270e+01
WFG1	10	4.635e+02	8.924e+01	1.008e+02	2.915e+02	5.137e+02	9.116e+01
WFG3	10	1.902e+03	8.662e+01	8.612e+01	2.802e+02	6.022e+02	9.028e+01

Table 11: Maximum Pareto Front Error

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	2.008e+01	1.195e+01	8.083e+02	1.765e+01	3.548e+02	4.912e+00
DTLZ3	5	1.079e+01	1.564e+00	2.545e+01	1.604e+00	1.546e+01	5.798e-01
WFG1	5	1.332e-01	1.763e-02	2.620e-01	2.042e-01	1.709e-01	1.588e-02
WFG3	5	1.583e-01	0.000e+00	9.601e-02	6.763e-02	1.139e-01	0.000e+00
DTLZ1	6	2.937e+02	5.789e+01	1.583e+03	5.168e+01	3.920e+02	7.665e+00
DTLZ3	6	1.045e+03	2.861e+02	1.825e+03	1.913e+02	7.048e+02	7.409e+01
WFG1	6	2.288e-01	1.619e-02	3.790e-01	3.086e-01	2.649e-01	1.372e-02
WFG3	6	1.690e-01	0.000e+00	1.090e-01	7.179e-02	9.973e-02	2.058e-03
DTLZ1	7	1.193e+02	4.205e+01	8.990e+02	9.095e+00	1.081e+02	2.998e+00
DTLZ3	7	1.138e+01	2.539e+00	1.768e+01	3.267e-01	4.447e+00	1.286e-01
WFG1	7	2.461e-01	1.443e-02	3.670e-01	2.775e-01	2.428e-01	1.545e-02
WFG3	7	1.630e-01	6.556e-04	1.017e-01	6.336e-02	7.499e-02	2.411e-03
DTLZ1	8	4.798e+02	2.375e+02	1.982e+03	4.991e+01	5.619e+02	8.178e+00
DTLZ3	8	1.458e+03	3.881e+02	2.152e+03	1.856e+02	9.085e+02	6.259e+01
WFG1	8	2.722e-01	1.486e-02	4.113e-01	3.155e-01	3.020e-01	1.039e-02
WFG3	8	1.499e-01	0.000e+00	9.124e-02	5.919e-02	6.697e-02	2.380e-03
DTLZ1	9	1.732e+02	1.234e+02	9.926e+02	1.264e+01	3.271e+02	1.976e+00
DTLZ3	9	7.820e+00	3.242e+00	1.899e+01	2.121e-01	6.489e+00	9.978e-02
WFG1	9	2.388e-01	1.108e-02	3.644e-01	2.316e-01	2.435e-01	7.929e-03
WFG3	9	1.516e-01	4.995e-04	8.803e-02	5.787e-02	8.046e-02	1.736e-03
DTLZ1	10	1.097e+02	1.138e+02	9.838e+02	8.148e+00	3.040e+02	2.231e+00
DTLZ3	10	6.727e+00	2.405e+00	1.556e+01	1.584e-01	5.933e+00	7.632e-02
WFG1	10	3.030e-01	1.372e-02	4.268e-01	2.544e-01	3.118e-01	9.250e-03
WFG3	10	1.468e-01	3.964e-04	7.328e-02	5.557e-02	6.889e-02	2.378e-03

Table 12: Average compute time in Maximum Pareto Front Error experiment

	m	eMOEA	NSGA-R	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	4.128e+01	5.408e+01	5.408e+01	2.401e+02	2.308e+02	5.522e+01
DTLZ3	5	5.676e+01	6.470e+01	5.944e+01	2.074e+02	2.982e+02	6.294e+01
WFG1	5	1.623e+02	8.048e+01	7.232e+01	2.239e+02	2.815e+02	7.082e+01
WFG3	5	9.397e+02	7.952e+01	6.154e+01	2.043e+02	3.174e+02	1.023e+02
DTLZ1	6	4.550e+01	5.556e+01	5.634e+01	1.924e+02	1.662e+02	5.686e+01
DTLZ3	6	9.168e+01	6.554e+01	6.418e+01	1.985e+02	1.787e+02	6.656e+01
WFG1	6	1.958e+02	7.512e+01	7.434e+01	2.072e+02	2.170e+02	7.650e+01
WFG3	6	1.136e+03	7.774e+01	6.724e+01	1.967e+02	2.406e+02	7.916e+01
DTLZ1	7	8.734e+01	6.188e+01	6.164e+01	2.453e+02	2.058e+02	6.204e+01
DTLZ3	7	1.622e+02	7.046e+01	7.170e+01	2.699e+02	1.812e+02	8.968e+01
WFG1	7	2.674e+02	8.156e+01	8.470e+01	2.574e+02	2.154e+02	8.036e+01
WFG3	7	1.461e+03	8.358e+01	7.426e+01	2.546e+02	2.334e+02	8.180e+01
DTLZ1	8	8.920e+01	6.054e+01	6.592e+01	2.443e+02	2.349e+02	6.252e+01
DTLZ3	8	2.360e+02	7.318e+01	7.644e+01	2.536e+02	2.555e+02	7.426e+01
WFG1	8	4.476e+02	7.960e+01	8.678e+01	2.612e+02	2.932e+02	8.164e+01
WFG3	8	1.482e+03	8.250e+01	7.690e+01	2.497e+02	3.244e+02	8.380e+01
DTLZ1	9	1.031e+02	6.208e+01	6.984e+01	2.514e+02	3.068e+02	6.554e+01
DTLZ3	9	3.043e+02	7.924e+01	8.222e+01	2.634e+02	3.368e+02	7.806e+01
WFG1	9	3.935e+02	8.856e+01	9.290e+01	2.700e+02	3.807e+02	8.676e+01
WFG3	9	1.660e+03	9.016e+01	8.028e+01	2.594e+02	4.347e+02	8.622e+01
DTLZ1	10	1.507e+02	6.436e+01	7.442e+01	2.728e+02	4.151e+02	6.830e+01
DTLZ3	10	3.933e+02	8.494e+01	8.852e+01	2.865e+02	4.593e+02	8.244e+01
WFG1	10	4.769e+02	9.296e+01	9.974e+01	2.904e+02	5.110e+02	9.182e+01
WFG3	10	1.875e+03	8.474e+01	8.594e+01	2.784e+02	6.013e+02	9.096e+01

often NSGA-G with random metric. It can be accepted for the trade-off between quality and computation time.

Study on the evaluation In the previous experiments, we survey algorithms with various problems and the constant number of max evaluation. This section selects a specific problem and shows the observation of algorithms while the process is running. In particular, we choose DTLZ3 problem with eight objectives, called DTLZ3-8. Besides, we focus on reducing the execution time of NSGAs algorithm. Hence, this section compares two versions of NSGA-G algorithms to others in NSGA class, such as NSGA-II and NSGA-III. Two versions of NSGA-G with Min point and Random metric are called NSGA-G and NSGA-R, respectively. The results in Fig. 10 and 11 show that two versions of NSGA-G are faster than others. Both their convergence and diversity are better than NSGA-II and NSGA-III.

In conclusion, NSGA-Gs often show better quality and faster execution time in most cases, such as DTLZs, WFGs. One main conclusion of these experiments is that NSGA-G with a Random metric is often the least expensive in terms of computation.

6 Conclusion

This paper is about medical data management in cloud federation. It introduces Dynamic Regression Algorithm (DREAM) as a part of MIDAS and on top of IReS, an open source platform for complex analytics work-flows executed over

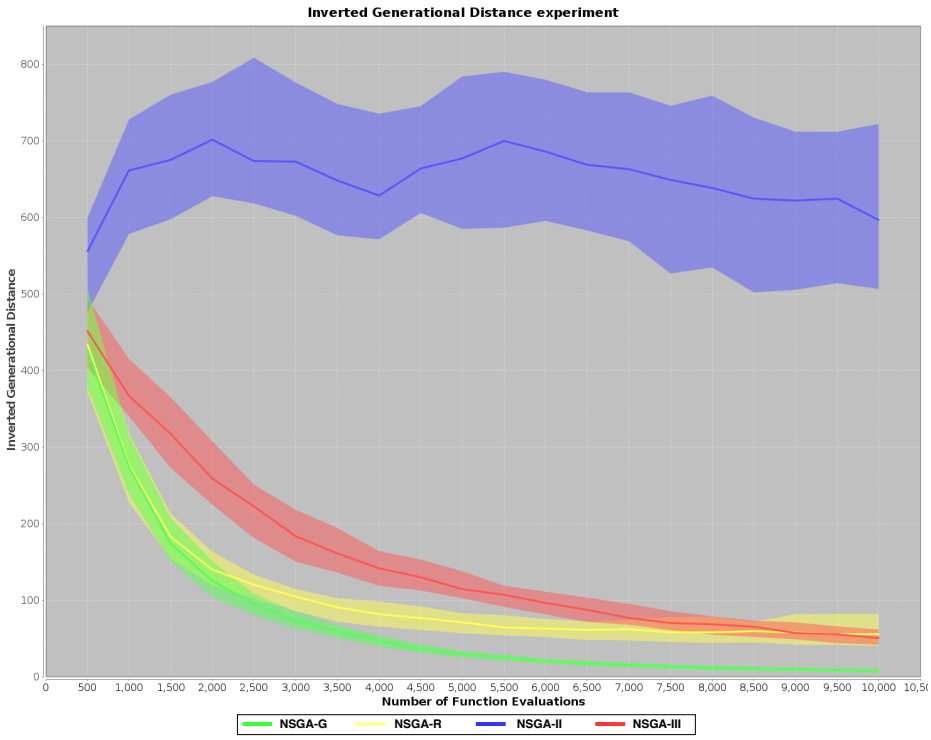

Fig. 10: Inverted Generational Distance of 4 algorithms with DTLZ3-8.

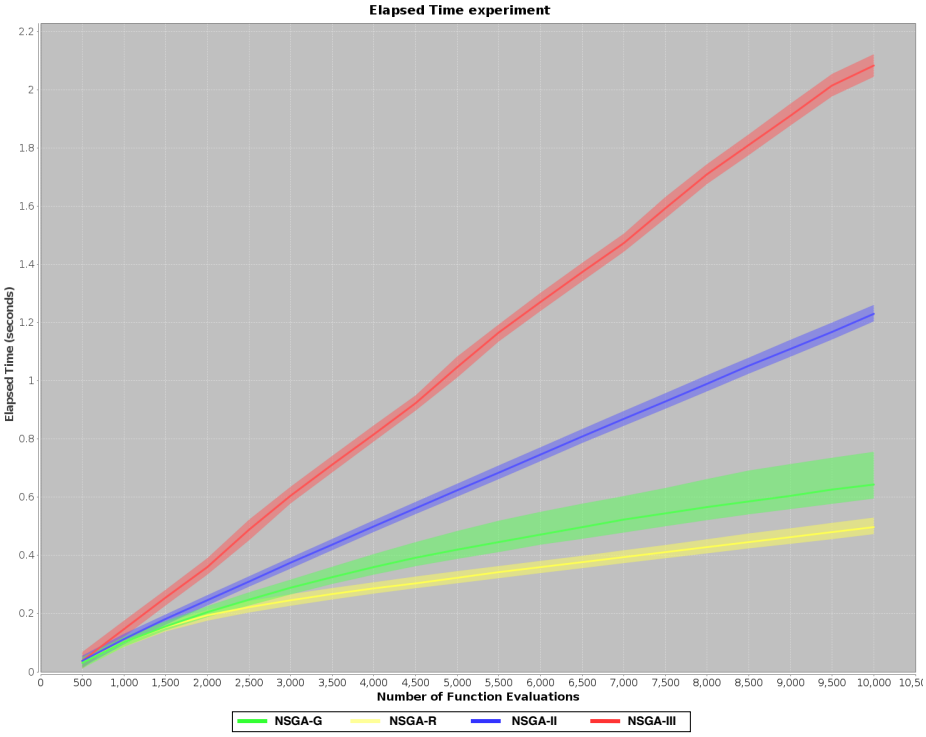

Fig. 11: Execution time of 4 algorithms with DTLZ3-8.

multi-engine environments. DREAM aims to address variance in a cloud federation and to provide accurate estimation for MOQP. Experiment results with DREAM and TPC-H benchmark are quite promising with respect to existing solutions. Further more, we introduce Non-dominated Sorting Algorithms based on Grid partitioning (NSGA-G) in searching and optimization MOOP. We validated NSGA-Gs with DTLZ, WFG test problems, and MOEA framework. The experiments show that NSGA-Gs often show better quality and faster execution time than other NSGAs in most cases, such as DTLZs, WFGs. One main conclusion of these experiments is that NSGA-G with a Random metric is often the least expensive in terms of computation.

In the future, we plan to validate our proposal with more cloud providers (and their associated pricing model and services) and data management systems. We will also define new strategies to choose QEPs in a Pareto Set. Further more, the size of population in each generation iterate is constant in many NSGAs. The suitable value of population size is still a question of NSGAs. Future works include a deeper study on the impact of the size of the population.

References

1. Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P.A., Carey, M.J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M.J., Gehrke, J., Haas, L.M., Halevy, A.Y., Hellerstein, J.M., Ioannidis, Y.E., Jagadish, H.V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J.F., Ramakrishnan, R., Markl, V., Olston, C., Ooi, B.C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., Widom: The Beckman report on database research, J. Commun. ACM, 59(2): pp. 92–99(2016)
2. Akdere, M., Çetintemel, U., Riondato, M., Upfal, E., Zdonik, S.B.: Learning-based Query Performance Modeling and Prediction, 2012 IEEE 28th International Conference on Data Engineering, Washington, DC, 2012, pp. 390-401.
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4): 50-58 (2010).
4. Bezerra, L.C.T., López-Ibáñez, M., Stützle T.: An Empirical Assessment of the Properties of Inverted Generational Distance on Multi- and Many-Objective Optimization, Evolutionary Multi-Criterion Optimization, Springer International Publishing, pp. 31-45, 2017.
5. Breiman, L.: Bagging predictors. Mach. Learn. 24(2): 123-140, Aug. 1996.
6. Bugiotti, F., Bursztyn, D., Deutsch, A., Ileana, I., Manolescu, I.: Invisible Glue: Scalable Self-Tuning Multi-Stores(2015), Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, 2015.
7. Cerf, R.: Asymptotic convergence of genetic algorithms, Advances in Applied Probability, Cambridge University Press, vol. 30, no. 2, pp. 521–550, 1998.
8. Chankong, V., Haimes, Y.Y.: Multiobjective decision making: theory and methodology, North-Holland series in system science and engineering, North Holland, 1983.
9. Coello, C.A.C., Cortés, N.C.: Solving Multiobjective Optimization Problems Using an Artificial Immune System. Genet Program Evolvable Mach 6, 163–190 (2005)
10. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary algorithms for solving multi-objective problems, Second Edition. Genetic and evolutionary computation series, Springer 2007, ISBN 978-0-387-33254-3, pp. I-XXI, 1-800

11. DeWitt, D.J., Halverson, A., Nehme, R., Shankar, S., Aguilar-Saborit, J., Avanes, A., Flasza, M., Gramling, J.: Split query processing in polybase. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). Association for Computing Machinery, New York, NY, USA, 1255–1266, 2013.
12. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1): 107-113. Jan. 2008.
13. Deb, K., Agrawal R.B.: Simulated Binary Crossover for Continuous Search Space, Complex Systems, vol. 9, pp. 1-34, 1994.
14. Deb, K., Jain, H.: An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints, in IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577-601, Aug. 2014.
15. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II, in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002.
16. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization, Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105-145, 2005.
17. Doka, K., Papailiou, N., Tsoumakos, D., Mantas, C., Koziris, N.: IReS: Intelligent, Multi-Engine Resource Scheduler for Big Data Analytics Workflows. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15). ACM, New York, NY, USA, 1451-1456.
18. Elmore, A., Duggan, J., Stonebraker, M., Balazinska, M., Cetintemel, U., Gadepally, V., Heer, J., Howe, B., Kepner, J., Kraska, T., Madden, S., Maier, D., Mattson, T., Papadopoulos, S., Parkhurst, J., Tatbul, N., Vartak, M., Zdonik, S.: A demonstration of the BigDAWG polystore system. Proc. VLDB Endow. 8, 12, 1908–1911, 2015.
19. Fard, H.M., Prodan, R., Barrionuevo, J.J.D., Fahringer, T.: A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), Ottawa, ON, 2012, pp. 300-309.
20. Fonseca, C.M., Fleming, P.J.: An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, vol. 3, no. 1, pp. 1-16, March 1995.
21. Ganapathi, A., Kuno, H., Dayal, U., Wiener, J.L., Fox, A., Jordan, M., Patterson, D.: Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. In: 2009 IEEE 25th International Conference on Data Engineering, Shanghai, pp. 592–603(2009)
22. Giannakouris, V., Papailiou, N., Tsoumakos, D., Koziris, N.: MuSQLE: Distributed SQL query execution over multiple engine environments, 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, pp. 452-461, 2016.
23. Glaßer, C., Reitwießner, C., Schmitz, H., Witek, M.: Approximability and Hardness in Multi-objective Optimization. In: Ferreira F., Löwe B., Mayordomo E., Mendes Gomes L. (eds) Programs, Proofs, Processes. CiE 2010. Lecture Notes in Computer Science, vol 6158. Springer, Berlin, Heidelberg, 2010.
24. Helff, F., Gruenwald, L., D'Orazio, L.: Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments, EDBT/ICDT Workshops, 2016.

25. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit, IEEE Transactions on Evolutionary Computation, vol. 10, pp. 447-506, num. 5, Oct 2006.
26. Ishibuchi, H., Masuda, H., Nojima, Y.: Sensitivity of performance evaluation results by inverted generational distance to reference points, IEEE Congress on Evolutionary Computation (CEC), pp. 1107-1114, Jul. 2016.
27. Jain, H., Deb, K.: An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach, in IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 602-622, Aug. 2014.
28. Karpathiotakis, M., Alagiannis, I., Ailamaki, A.: Fast queries over heterogeneous data through engine customization. Proc. VLDB Endow. 9, 12, 972–983, 2016.
29. Khan, S.A., Rehman, S.: Iterative non-deterministic algorithms in on-shore wind farm design: A brief survey, Renewable and Sustainable Energy Reviews, 19, 370 - 384, 2013.
30. Kllapi, H., Sitaridi, E., Tsangaris, M.M. , Ioannidis, Y.: Schedule optimization for data processing flows on the cloud, Proceedings of the 2011 international conference on Management of data - SIGMOD '11, pp. 289, 2011.
31. Knowles, J., Corne, D.: The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation, Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 98-105 Vol. 1.
32. Kolev, B., Bondiombouy, C., Valduriez, P., Jimenez-Peris, R., Pau, R., Pereira, J.: The CloudMdsQL Multistore System. In Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16). Association for Computing Machinery, New York, NY, USA, 2113–2116, 2016.
33. Kolev, B., Valduriez, P., Bondiombouy, C., Jiménez-Peris, R., Pau, R., Pereira, J.: CloudMdsQL: querying heterogeneous cloud data stores with a common language. Distrib. Parallel Databases 34, 4, 463–503, 2016.
34. Köppen, M., Yoshida, K.: Substitute Distance Assignments in NSGA-II for Handling Many-objective Optimization Problems, pp. 727-741, Jan. 2006.
35. Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., Kunze, M.: Cloud federation. The Second International Conference on Cloud Computing, GRIDs, and Virtualization. 32-38. 2011.
36. Le, T.-D., Kantere, V., D'Orazio, L.: An efficient multi-objective genetic algorithm for cloud computing: NSGA-G, IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018, pp. 3883-3888, 2018.
37. Le, T.-D., Kantere, V., D'Orazio, L.: Dynamic Estimation for Medical Data Management in a Cloud Federation, Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019.
38. LeFevre, J., Sankaranarayanan, J., Hacigümüs, H., Tatemura, J., Polyzotis, N., Carey, M.J.: MISO: souping up big data query processing with a multistore system. SIGMOD Conference 2014: 1591-1602, 2014.
39. Nykiel, T., Potamias, M., Mishra, C., Kollios, G., Koudas, N.: MRShare: Sharing Across Multiple Queries in MapReduce. PVLDB, 3, 494-505. 2010.
40. Papakonstantinou, Y.: Polystore Query Rewriting: The Challenges of Variety, EDBT/ICDT Workshops, 2016.
41. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley & Sons, Inc., New York, NY, USA. 1987.

42. Sidhanta, S., Golab, W., Mukhopadhyay, S.: OptEx: A Deadline-Aware Cost Optimization Model for Spark, 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, 2016, pp. 193-202.
43. Soong, T.T.: Fundamentals of probability and statistics for engineers, John Wiley & Sons, 2004.
44. Srinivas, N., Deb, K.: Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, vol. 2, no. 3, pp. 221-248, Sept. 1994.
45. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., Murthy, R. (2010): Hive - a petabyte scale data warehouse using Hadoop. 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), 996-1005.
46. Tozer, S., Brecht, T., Aboulnaga, A.: Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads, 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), Long Beach, CA, 2010, pp. 397-408.
47. Trummer, L., Koch, C.: Multi-objective parametric query optimization. Commun. ACM 60(10), 81-89 (2017).
48. Veldhuizen, D.A.V.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.
49. Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Computation and Convergence to a Pareto Front, Late Breaking Papers at the Genetic Programming 1998 Conference, pp. 221-228, 1998.
50. Wu, W., Chi, Y., Zhu, S., Tatemura, J., Hacigümüs, H., Naughton, J.F.: Predicting query execution time: Are optimizer cost models really unusable?, 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, QLD, 2013, pp. 1081-1092.
51. Xiong, P., Chi, Y., Zhu, S., Tatemura, J., Pu, C., HacigümüŞ, H.: ActiveSLA: A profit-oriented admission control framework for Database-as-a-Service providers. Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC 2011.
52. Yen, G.G., He, Z.: Performance Metrics Ensemble for Multiobjective Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, 2013.
53. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712-731, Dec. 2007.
54. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK-Report. 103, 2001.
55. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Fonseca, V.G.D.: Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation, vol. 7, no. 2, pp. 117-132, April 2003.