



HAL
open science

Data Management in a Cloud Federation

Trung Dung Le, Ecole Doctorale, Karine Zeitouni

► **To cite this version:**

Trung Dung Le, Ecole Doctorale, Karine Zeitouni. Data Management in a Cloud Federation. Computer Science [cs]. Rennes 1, 2019. English. NNT: . tel-02931831v2

HAL Id: tel-02931831

<https://hal.science/tel-02931831v2>

Submitted on 18 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE UNIVERSITE BRETAGNE LOIRE

Ecole Doctorale N°601
*Mathématique et Sciences et Technologies
de l'Information et de la Communication*
Spécialité : *Informatique*
Par

« **Trung-Dung LE** »

« **Gestion de masses de données dans une fédération
de nuages informatiques** »

Thèse présentée et soutenue à LANNION , le 11 juillet 2019
Unité de recherche : IRISA

Rapporteurs avant soutenance :

Abdelkader Hameurlain Professeur à Université Paul Sabatier, IRIT
Dimitris Kotzinos Professeur à Université de Cergy-Pontoise, ENSEA, CNRS

Composition du jury :

Président :

Examineurs : Karine Zeitouni Professeur à University of Versailles St Quentin, DAVID Lab
 Tassadit Bouadi Maître de Conférences à Université de Rennes 1, CNRS, IRISA
 Abdelkader Hameurlain Professeur à Université Paul Sabatier, IRIT
 Dimitris Kotzinos Professeur à Université de Cergy-Pontoise, ENSEA, CNRS

Dir. de thèse : Laurent d'Orazio Professeur à l'Université de Rennes 1, CNRS, IRISA

Co-dir. de thèse : Verena Kantere Professeur à l'Université de Ottawa

AUTHOR

Trung-Dung LE - Ph.D. student.
Email : trung-dung.le@irisa.fr
dunglt@wru.vn



Trung-Dung LE was born in Hanoi, Vietnam in 1980. He graduated his B.S degree in Electrical Engineering at Hanoi University of Science and Technology, Vietnam in 2003. In 2005, he received his M.S degree in the same university. At the end of 2005, he started to begin teaching at Thuy Loi University in the Electrical Engineering department. He started his Ph.D. degree in Computer Science at CNRS, LIMOS UMR 6158, Clermont Auvergne University, France In September of 2015. In November of 2016, he changed to the Doctorate School of Bretagne University. His research interests include Cloud Computing, Multi-Objective Optimization, Database System, Big Data, and Machine Learning.

DEDICATION

To Mom and Dad, there is no word adequately to thank you for everything you've done for me.

When I was young, Mom always showed as a big friend, listening and giving me helpful advice. She is the first person opening the sky of knowledge which is my passion. Dad protected, loved and raised me with the infinite love.

For all of my love who gave beautiful memories, my grandmother Hong and my sister Chung.

And to Quang Minh, Nam Phong, Quoc Trung, Hoang Anh, Hong Truong, Quoc Khanh, Le Cuong, Tuan Khanh, Quoc Thinh, Viet Ha, Hoang Linh, Duong Hung, class 9A - Ngo Quyen school, class 12A - Thang Long school, thanks for all of the beautiful memories of growing up, and for your continued support and encouragement.

For all of my loved ones who've gone on to a better life, especially my mother and father in law, my grandfather, grandmother - you are always close in heart.

And to my Ph.D. friends, Cong Danh, Sy Dat, Anh Duy, Van Dung, Van Giang, Mai Thanh, Hugues especial Xuan Chien, my very first reader for all my publications, thank you for all of the feedback.

Finally, this dedication would not be complete without an exceptional thank you to Binh, my wife, and Dat, An, my children who are always loving, living beside me.

God bless you !

DECLARATION

This thesis has been completed by Trung-Dung LE under the supervision of Prof. Laurent d’Orazio and Prof. Verena Kantere. It has not been submitted for any other degree or professional qualification. I declare that the work presented in this thesis is entirely my own except where indicated by full references.

SIGNATURE



ACKNOWLEDGEMENTS

First of all, I would like to thank the Ministry of Education and Training of Vietnam for offering me an International Postgraduate Research Scholarship (Project 911) to support my doctoral program in France. I would also like to express my gratitude to Thuy Loi University for allowing me to receive and retain this scholarship. Besides, I would like to thank all the financial support from IRISA laboratory and technical help from Shaman team in Lannion.

I would like to show deep gratitude to my Ph.D. supervisors, Prof. Laurent d’Orazio and Prof. Verena Kantere for their guidance and advice. Prof. Laurent d’Orazio always beside me to discuss and show me the excellent knowledge of cloud computing and database research. He often showed the new level of research we should deal with. The new problem and the direction are often opened through our discussion. Many thanks to Prof. Verena Kantere for her supervision. In every meeting, she supported the idea and the implementation so much. She helped the research and provided many valuable advice on the works. Once again, I would like to thank all of my supervisors who supported, guided, advised and reviewed my job through all of the phases.

Also, I would like to thank Frédéric Gaudet for providing a private cloud in Galactica platform in ISIMA laboratory.

Living and studying in a foreign country is not easy. The challenges are met through day by day. Many thanks to Cong-Danh Nguyen in Blaise Pascal University - Clermont II for providing valuable DICOM dataset.

Finally, I would like to thank my big family for always loving, living beside me in difficult moments and motivating me.

Lannion, 2019



Trung-Dung Le

RESUMÉ

Les fédérations de nuages (cloud federations) peuvent être considérées comme une avancée majeure de l'informatique en nuage (cloud computing), en particulier pour le domaine médical. En effet, le partage de données médicales permet d'améliorer la qualité des soins. La fédération de ressources rend possible l'accès à toutes les informations, même sur une personne mobile, avec des données hospitalières distribuées sur différents établissements. De plus, le volume d'information disponible étant plus important et sur de plus nombreux patients, les statistiques obtenus sont alors plus précises.

Les données médicales sont généralement conformes à la norme DICOM (Digital Imaging and Communications in Medicine). Les fichiers DICOM peuvent être stockés sur différentes plates-formes, telles qu'Amazon, Microsoft, Google Cloud, etc. La gestion des fichiers, y compris le partage et le traitement, sur ces plates-formes, suit un modèle de paiement à l'utilisation, selon des modèles de prix distincts et en s'appuyant sur divers systèmes de gestion de données (systèmes de gestion de données relationnelles ou SGBD ou systèmes NoSQL). En outre, les données DICOM peuvent être structurées en lignes ou colonnes [125, 19, 134, 132] ou selon une approche hybride (ligne-colonne) [63, 46, 101]. En conséquence, la gestion des données médicales dans des fédérations de nuages soulève des problèmes d'optimisation multi-objectifs (MOOP - Multi-Objective Optimization Problems) pour (1) le traitement des requêtes et (2) le stockage des données, selon les préférences des utilisateurs, telles que le temps de réponse, le coût monétaire, la qualité, etc. Ces problèmes sont complexes à traiter en raison de la variabilité de l'environnement (liée à la virtualisation, aux communications à grande échelle, etc.).

Les données médicales distribuées dans les fédérations de nuage amènent à intégrer des données de divers systèmes, tels que Hive Data Warehouse [128], PostgreSQL [133], etc. Il existe différents outils pour gérer les données dans plusieurs moteurs de base de données [83, 109, 144, 42]. De tels outils prennent en compte l'optimisation mais se concentrent sur un problème d'optimisation mono-objectif (SOOP - Single-Objective Optimization Problem), tel que la minimisation des transferts de

données. L'outil de gestion Intelligent Resource Scheduler (IReS) [42] constitue une exception à cette règle. Cette plate-forme de sources libres aborde le MOOP dans divers systèmes en combinant plusieurs objectifs en une valeur scalaire. Ce problème d'optimisation à objectif unique ne peut néanmoins pas représenter correctement les problèmes d'optimisation multi-objectifs [60]. Dans ce contexte, la construction d'un système de gestion des données médicales reposant sur le protocole MOOP dans une fédération de nuages constitue un problème majeur.

De plus, dans un environnement variable comme une fédération de nuages avec plusieurs systèmes de gestion de données, il est nécessaire de proposer une approche pour estimer les coûts considérés au sein du MOOP, tels que le temps de réponse, le coût monétaire, les transferts de données, la qualité, etc. en fonction des machines physiques, de la charge et des communications à grande échelle par exemple. Il existe deux classes de techniques de modélisation des coûts : avec et sans algorithmes d'apprentissage automatique. La première classe est limitée d'une part à des systèmes spécifiques, tels que MapReduce [36], PostgreSQL [153], Spark [135], etc., et d'autre part à un seul objectif, le temps d'exécution. La seconde classe nécessite souvent l'ensemble de l'historique des données pour construire un modèle de coût [154, 141, 4, 55]. Cela peut entraîner l'utilisation d'informations périmées. Dans ce contexte, le problème est de savoir comment estimer des valeurs précises pour un MOOP en utilisant des historiques de données pertinents dans une fédération de nuages.

De plus, les MOOPs peuvent être abordés à la fois pour le traitement des requêtes et le stockage des données de la gestion des données médicales dans une fédération de nuages. Un MOOP peut être résolu à l'aide d'algorithmes d'optimisation multi-objectifs ou du modèle de somme pondérée (WSM) [67] ou par la conversion en un problème SOOP (Single-Object Optimization Problem). Cependant, les SOOP ne peuvent pas représenter correctement les problèmes d'optimisation multi-objectifs [60]. De plus, les algorithmes d'optimisation multi-objectifs peuvent être sélectionnés en raison de leurs avantages par rapport à WSM. La solution optimale du WSM pourrait ne pas être acceptable, en raison d'une mauvaise définition des coefficients [50]. Les recherches effectuées [75] prouvent qu'un léger changement des poids peut entraîner des changements importants dans les vecteurs objectifs et que des poids très différents peuvent produire des vecteurs objectifs presque similaires. De plus, si le WSM change, un nouveau processus d'optimisation sera requis. Les MOOPs conduisent également à la recherche de solutions par des techniques de dominance de Pareto.

Cependant, il est souvent impossible de générer un front de Pareto optimal [160]. Cela conduit à trouver une solution approximative optimale par les techniques de dominance de Pareto. EMO (Evolutionary Multi-Objective Optimization) est une approche bien connue pour résoudre la grande complexité d'un MOOP. Parmi les approches EMO, les algorithmes NSGA (Nondominated Sorting Algorithms) [40, 37] ont une complexité de calcul inférieure à celle des autres approches EMO [40]. Ils garantissent le maintien de la diversité des solutions. Certains algorithmes génétiques multi-objectifs, NSGA-II [40] et SPEA-II [161] utilisent des distances d'encombrement pour maintenir la diversité. Le maintien de la diversité est une question essentielle, car les algorithmes évolutionnistes doivent pouvoir agrandir autant de régions que possible. Cependant, la complexité reste élevée, alors que la diversité ne peut être préservée avec plus de deux objectifs [76]. Zhang et Li [158] ont proposé MOEA/D pour maintenir la diversité avec plus de trois objectifs. MOEA/D utilise une approche de décomposition pour diviser plusieurs objectifs en divers sous-problèmes d'optimisation à objectif unique et résoudre jusqu'à quatre objectifs [118]. En outre, Deb et Jain [39] ont proposé la NSGA-III, qui utilise un ensemble de directions de référence pour guider le processus de recherche. Cependant, cet algorithme présente toujours une complexité de calcul élevée.

En outre, des travaux récents sur le stockage de données ont été proposés pour stocker les données afin d'optimiser la configuration des données hybrides. Cependant, HYRISE [63] et SAP HANA [46] ne tiennent pas compte du volume élevé et du caractère clairsemé des données DICOM. En outre, le modèle de paiement à la consommation mène à un MOOP pour trouver une configuration du stockage des données en fonction des préférences des utilisateurs en matière de temps de réponse, de coût monétaire, de qualité, etc. De plus, une approche automatique produisant des configurations de stockage de données pour les données DICOM dans [97] soulève un problème de MOOP dans les nuages. Les auteurs ont affirmé que le nombre de solutions candidates dans le MOOP était important, mais ils n'ont donné aucune méthode pour trouver les configurations de données hybrides optimales. Dans le contexte de MOOP, tant dans l'optimisation du traitement des requêtes que dans la configuration des données DICOM dans une fédération de nuages, le problème est de savoir comment optimiser un problème à objectifs multiples avec un algorithme à faible complexité de calcul.

Nos travaux se portent sur les fédérations de nuage dans le domaine médical.

Cette fédération comprend divers sites qui stockent, gèrent et partagent des données médicales. Dans chaque site, les données peuvent être stockées dans une variété de systèmes de gestion de données, tels que les systèmes en lignes, les systèmes en colonnes, les systèmes lignes et colonnes, et gérées par différents SGBD, tels que Hive, PostgreSQL, Spark, etc. Par conséquent, la fédération de nuages doit fonctionner avec divers moteurs de base de données et pour différentes structurations. En outre, la variabilité au sein d'une fédération en raison de la virtualisation, des communications à grande échelle, etc., nécessite l'utilisation efficace des historiques de données dans la construction du modèle de coût de construction. En outre, la gestion des données médicales dans des fédérations de nuages soulève également des problèmes d'optimisation multi-objectifs (MOOP) pour le traitement des requêtes et le stockage des données. Les MOOPs conduisent à la recherche de solutions par des techniques de domination de Pareto. Cependant, il est souvent impossible de générer un front de Pareto optimal en raison de sa grande complexité. Ainsi, l'important espace de candidats aux MOOPs conduit à la nécessité de trouver un ensemble approximatif de solutions de Pareto optimales. Par conséquent, une approche alternative pour trouver l'ensemble de optimal de Pareto, cherchant des approximations (ensemble de solutions proches du front optimal) est souvent utilisée. En conséquence, les MOOPs dans une fédération de cloud nécessitent une approche puissante pour trouver un ensemble approximatif de Pareto optimal.

Pour résoudre ces problèmes, nous proposons MIDAS (Medical system on cloud federAtionS), un système médical sur les fédérations de groupes. Un aperçu de MIDAS est présenté à la figure 1. Nous proposons un algorithme d'estimation des valeurs de coût dans au sein d'une fédération, appelé algorithme de régression multiple linéaire dynamique (DREAM). Cette approche permet de s'adapter à la variabilité de l'environnement en modifiant la taille des données et afin d'éviter d'utiliser des informations expirées. La figure 2 montre comment utiliser les données historiques de DREAM dans notre système. Nous proposons ensuite un algorithme génétique de tri non dominé pour résoudre le problème de la recherche et de l'optimisation de MOOPs dans un nuage informatique. Cet algorithme est appelé NSGA-G (Non-dominated Sorting Genetic Algorithm based Grid partitioning). NSGA-G est appliqué à un optimiseur Multi-Objectif, comme illustré par la figure 3. Troisièmement, les algorithmes ci-dessus sont appliqués à des MOOPs dans une fédération, notamment pour exécuter des requêtes et rechercher une configuration de données DICOM. Plus particulièrement, NSGA-G

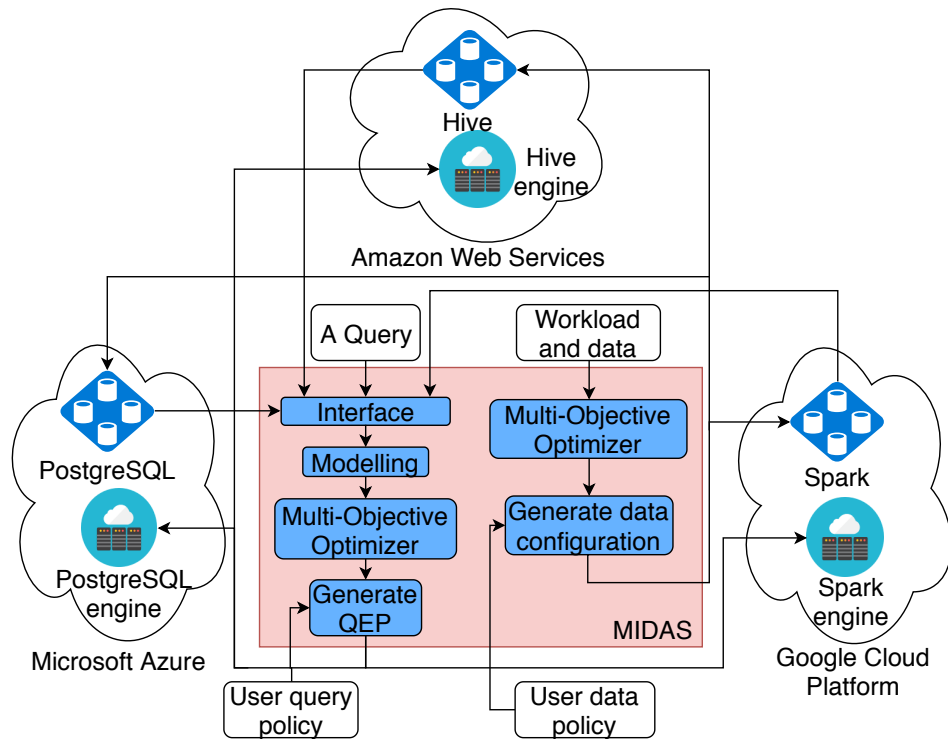


FIGURE 1 – Un aperçu de MIDAS.

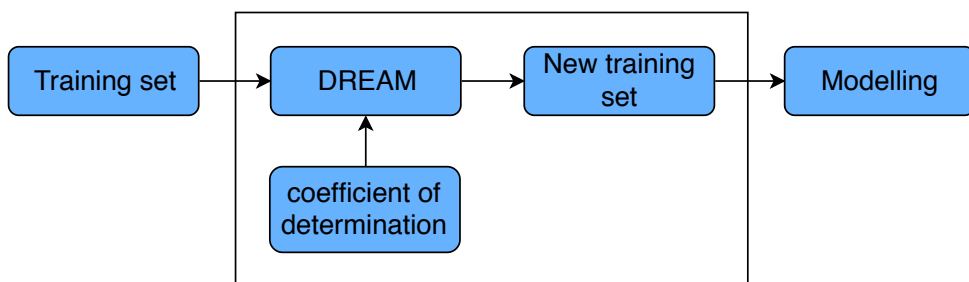


FIGURE 2 – Utiliser DREAM pour construire des modèles de coûts.

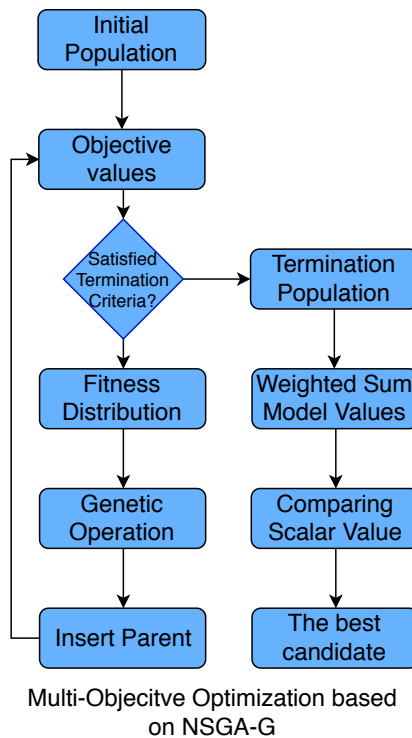


FIGURE 3 – Vue d’ensemble de l’optimisation multi-objectifs basée sur NSGA-G.

est utilisé pour trouver la meilleure structuration des données.

Les expériences réalisées valident DREAM, NSGA-G avec divers problèmes de test et jeux de données. DREAM est comparé selon la précision de l’estimation des valeurs de coût par rapport à d’autres algorithmes d’apprentissage automatique. Dans ces expériences, nous utilisons notamment un jeu de données de référence TPC-H pour valider DREAM. La qualité de la NSGA-G est comparée à celle des autres NSGAs présentant de nombreux problèmes dans le cadre du MOEA. Dans l’expérience NSGA-G, nous utilisons les problèmes DTLZ et WFG pour comparer la qualité de divers algorithmes génétiques à objectifs multiples. Le jeu de données DICOM est également expérimenté par NSGA-G dans le test de solution optimale. Enfin, MIDAS est validé à l’aide d’un nuage privé, la plate-forme Galactica, avec certains moteurs de base de données, tels que Hive, PostgreSQL, Spark. Les résultats expérimentaux montrent les bonnes qualités de nos solutions d’estimation et d’optimisation des MOOPs dans une fédération de nuages. De plus, notre algorithme trouve la solution optimale approximative dans un problème de configuration de données DICOM hybride.

TABLE OF CONTENTS

List of Abbreviations	XVIII
List of Figures	XX
List of Tables	XXII
1 Introduction	1
1.1 Context and problems	1
1.2 Related work	3
1.3 MIDAS	5
1.4 Outline	8
I State of the art about cloud federation	11
2 Cloud Federations	12
2.1 Introduction	12
2.2 Definitions	13
2.2.1 Motivating Example	13
2.2.2 Big data Management System	15
2.2.3 Cloud Computing	17
2.2.4 Cloud Federation	18
2.3 Resource Management	19
2.3.1 Virtualization	19
2.3.2 Partitioning	20
2.4 Data Management	21
2.4.1 Homogeneous system	21
2.4.2 Heterogeneous systems	22
2.5 Conclusion	25

TABLE OF CONTENTS

3	Optimization of medical data management	29
3.1	Introduction	29
3.2	Medical Data management	30
3.2.1	DICOM	30
3.2.2	Data Model	32
3.2.3	Hybrid data storage configuration	34
3.2.4	Vertical Partitioning	36
3.3	Search and Optimization	37
3.3.1	Single Objective Optimization	41
3.3.2	Multiple Objective Optimization	44
3.4	Conclusion	45
4	Multi-Objective Optimization	47
4.1	Introduction	47
4.2	Pareto set	48
4.3	Multiple Linear Regression	49
4.3.1	Linear Regression	50
4.3.2	Multiple Linear Regression	53
4.4	Non-dominated Sorting Genetic Algorithm	57
4.4.1	NSGA process	57
4.4.2	Application	60
4.5	Conclusion	61
II	Techniques for cloud federation	63
5	Dynamic Regression Algorithm	64
5.1	Introduction	64
5.2	Problem	64
5.3	DREAM	67
5.3.1	Coefficient of determination	67
5.3.2	Cost Value Estmation	69
5.3.3	Optimization	70
5.4	Conclusion	72

6	Non-dominated Sorting Genetic Algorithm based on Grid partitioning	75
6.1	Introduction	75
6.2	NSGA-G	76
6.2.1	Main process	76
6.2.2	Non-Dominated Sorting	76
6.2.3	Filter front process	77
6.3	Discussion	80
6.3.1	Convergence	80
6.3.2	Diversity	80
6.3.3	Computation	80
6.4	Selecting the size of grid	81
6.4.1	Simple front group	81
6.4.2	Max front group	82
6.5	Conclusion	82
7	Hybrid data storage configuration in cloud federation	85
7.1	Introduction	85
7.2	Medical system on cloud federation	86
7.2.1	MIDAS	86
7.2.2	IRES	86
7.2.3	Hybrid data storage configuration	87
7.2.4	Validation	88
7.3	Hybrid data storage configuration	90
7.3.1	Two phases of generating data storage configuration	90
7.3.2	Implementation	95
7.4	Optimizing data storage configuration	104
7.4.1	Finding Pareto configuration set	104
7.4.2	Finding the best configuration	104
7.5	Conclusion	105
III	Implementation of proposals and validations	107
8	Performance validation	108
8.1	Introduction	108

TABLE OF CONTENTS

8.2	A medical cloud federation	109
8.2.1	DICOM	109
8.2.2	Validation	109
8.3	DREAM	110
8.3.1	Implementation	110
8.3.2	Experiments	110
8.3.3	Results	112
8.4	NSGA-G	113
8.4.1	Validation on DTLZ test problems	113
8.4.2	Hybrid data storage configuration	127
8.5	Conclusion	134
9	Conclusion and Future work	137
9.1	Introduction	137
9.2	Summary and Conclusion	137
9.2.1	Existing solutions	138
9.2.2	Estimating in Multi-Objective Optimization Problem	139
9.2.3	Multi-Objective Evolutionary Algorithm	140
9.2.4	Optimizing medical data storage configuration	140
9.3	Future Works	141
9.3.1	Estimation	141
9.3.2	Searching and optimization	141
9.3.3	Hybrid data storage configuration	142
	Bibliography	145

LIST OF ABBREVIATIONS

- AASM** Attribute Access Similarity Matrix
- ADSM** Attribute Density Similarity Matrix
- Amazon RDS** Amazon Relational Database Service
- API** Application Programming Interface
- AUM** Attribute Usage Matrix
- BigDAWG** Big Data Analytics Working Group
- CloudMdsQL** Cloud Multidatastore Query Language
- DBMS** Database Management System
- DICOM** Digital Imaging and Communications in Medicine
- DREAM** Dynamic Regression Algorithm
- EA** Evolutionary Algorithm
- EAs** Evolutionary Algorithms
- EC** Evolutionary Computation
- EMO** Evolutionary Multi-Objective Optimization
- GICTF** Global Inter-Cloud Technology Forum
- HDFS** Hadoop Distributed File System
- HSM** Hybrid Similarity Matrix
- IReS** Intelligent Resource Scheduler
- JDBC** Java Database Connectivity
- MO** Multi-Objective
- MOEA** Multiple Objective Evolutionary Algorithm
- MOO** Multi-Objective Optimization
- MOOP** Multi-Objective Optimization Problem

List of Abbreviations

- MOP** Multi-Objective Problem
- MOQO** Multi-Objective Query Optimization
- MOQP** Multi-Objective Query Processing
- MuSQLE** Distributed SQL Query Execution Over Multiple Engine Environments
- NoSQL** Non Structured Query Language
- NSGA** Non-dominated Sorting Genetic Algorithm
- NSGA-G** Non-dominated Sorting Genetic Algorithm based on Grid Partitioning
- ODBC** Open Database Connectivity
- OLAP** Online analytical processing
- OLTP** Online transaction processing
- PAES** Pareto Archived Evolution Strategy
- QEP** Query Execution Plan
- R^2 Coefficient of Determination
- RDBMS** Relational DataBase Management System
- SA** Simulated Annealing
- SO** Single-Objective
- SOOP** Single-Objective Optimization Problem
- SOP** Single-Objective Problem
- SPEA** Strength Pareto Evolutionary Algorithm
- SQL** Structured Query Language

LIST OF FIGURES

1	Un aperçu de MIDAS.	XI
2	Utiliser DREAM pour construire des modèles de coûts.	XI
3	Vue d'ensemble de l'optimisation multi-objectifs basée sur NSGA-G. . .	XII
1.1	An overview of MIDAS	6
1.2	Using DREAM to build cost models.	6
1.3	An overview of Multi-Objective Optimization based on NSGA-G.	7
2.1	Motivating Example on using cloud federation.	14
2.2	A simple cluster.	20
2.3	FORWARD Query Processing [109].	26
2.4	BigDAWG Architecture [45].	27
2.5	MuSQLE system architecture [57].	27
2.6	Architecture of IReS [42].	28
3.1	Difference configurations of the table T	33
3.2	HYRISE architecture [63].	36
4.1	NSGA-II and NSGA-III procedure [40, 39].	57
4.2	An example of using the crowing distance in NSGA-II.	59
5.1	DREAM module.	68
5.2	Comparing two MOQP approaches	71
6.1	An example of using Grid points.	78
6.2	A simple front group.	82
6.3	A max front group.	83
7.1	An example of MIDAS system	87
7.2	The horizontal table T	89
8.1	Implementation of MIDAS, DREAM and Multi-Objective Optimizer. . . .	111
8.2	Inverted Generational Distance of 4 algorithms with DTLZ3_8.	125

8.3 Execution time of 4 algorithms with DTLZ3_8. 126

LIST OF TABLES

2.1	Example of instances pricing.	15
2.2	Multiple Objectives for Query Execution Plans	15
2.3	Advantage and disadvantage of recent researches.	25
3.1	Example of real DICOM data set [97].	31
3.2	Example of extracted DICOM data set [97].	31
3.3	Frequency of Queries in Workload W.	32
4.1	Multiple Objectives for Query Execution Plans	49
5.1	Using MLR in different size of dataset.	68
7.1	AUM and frequencies	89
7.2	AUM and frequencies	90
8.1	Comparison of mean relative error with 100MiB TPC-H dataset.	112
8.2	Comparison of mean relative error with 1GiB TPC-H dataset.	113
8.3	Generational Distance	115
8.4	Average compute time (seconds) in Generational Distance experiment .	116
8.5	Inverted Generational Distance	116
8.6	Average compute time (seconds) in Inverted Generational Distance ex- periment	117
8.7	Maximum Pareto Front Error	117
8.8	Average compute time (seconds) in Maximum Pareto Front Error exper- iment	118
8.9	Generational Distance	119
8.10	Average compute time in Generational Distance experiment	120
8.11	Inverted Generational Distance	121
8.12	Average compute time in Inverted Generational Distance experiment . .	122
8.13	Maximum Pareto Front Error	123
8.14	Average compute time in Maximum Pareto Front Error experiment . . .	124

List of Tables

8.15 Example of real DICOM data set.	127
8.16 Example of extracted DICOM data set.	127
8.17 Frequency of Queries in Workload W_P	129
8.18 Attribute Usage Matrix of Patient table.	130
8.19 Frequency of Queries in Workload W_S	131
8.20 Attribute Usage Matrix of Study table.	131
8.21 Frequency of Queries in Workload W_{Ge}	132
8.22 Attribute Usage Matrix of GeneralInfoTable.	132
8.23 Frequency of Queries in Workload W_{Seq}	133
8.24 Attribute Usage Matrix of SequenceAttributes.	133
8.25 Generational Distance.	134
8.26 Inverted Generational Distance.	134
8.27 Maximum Pareto Front Error	134
8.28 The execution time of NSGAs with DICOM.	134

INTRODUCTION

1.1 Context and problems

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

Medical data usually conform to the Digital Imaging and Communications in Medicine (DICOM) standard. DICOM has been widely used by medical institutions, hospitals, diagnostic centers and analysis laboratories. DICOM files can be stored on different cloud platforms, such as Amazon, Microsoft, Google Cloud, etc. The management of the files, including sharing and processing, on such platforms, follows the pay-as-you-go model, according to distinct pricing models and relying on various systems. In addition, DICOM data can be structured following traditional (row or column) [125, 19, 134, 132] or hybrid (row-column) [63, 46, 101] data storages. As a consequence, medical data management in cloud federations raises Multi-Objective Optimization Problems (MOOPs) for (1) query processing and (2) data storage, according to users preferences, related to various measures, such as response time, monetary cost, qualities, etc. These problems are complex to address because of the variability of the environment (due to virtualization, large-scale communications, etc.).

First, distributed medical data in cloud federations leads to integrate data from various systems, such as Hive data warehouse [128], PostgreSQL [133], etc. There are various tools for managing data in multiple database engines [83, 109, 144, 42]. Such tools perform optimization but focus on a Single-Objective Optimization Problem (SOOP), such as minimizing data transfers. An exception to this is the management tool Intelligent Resource Scheduler (IReS) [42]. This open-source platform solves Multi-Objective Optimization Problem (MOOP) in various systems by combining multi-

ple objectives into a scalar value. This Single-Objective Optimization Problem cannot adequately represent Multi-Objective Optimization Problems [60]. In this context, the first challenging problem is how to build a medical data management system relying on MOOP in a cloud federation.

Second, in a variable environment like a cloud federation with various database systems, we should build a model to estimate the objective values of the cost model for the MOOP. It also depends on the variety of physical machines, load, and wide-range communications. There are two classes of cost modeling, without and with machine learning algorithms. The first class is limited into a the specific system, such as MapReduce [36], PostgreSQL [153], Spark [135], etc., and a Single-Objective, i.e the execution time. The second class often requires the entire historical data to build a cost model [154, 141, 4, 55]. It may lead to the use of the expired information. In this context, the second challenging problem is how to estimate accurate values for MOOP by using efficient historical data in a cloud federation.

Third, MOOP could be solved by Multi-Objective Optimization algorithms or the Weighted Sum Model (WSM) [67] or converting to a SOOP. However, SOOPs cannot adequately represent Multi-Objective Optimization Problems [60]. Besides, 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 [50]. The research in [75] 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. Furthermore, MOOPs leads to find solutions by Pareto dominance techniques. However, generating a Pareto-optimal front is often infeasible due to high complexity [160]. It leads to finding an approximate optimal solution by Pareto dominance techniques. A well known approach to solve the high complexity of MOOP is Evolutionary Multi-Objective Optimization (EMO). Among EMO approaches, Non-dominated Sorting Genetic Algorithms (NSGAs) [40, 37] have lower computational complexity than other EMO approaches [40]. However, this algorithm still has high computational complexity.

In addition, some recent works [63, 46, 97, 101] on data storage have been proposed to optimize the hybrid data storage configuration. However, they do not consider the high volume and sparsity of DICOM data [63, 46], or do not give any method to find the optimal hybrid data storage configurations [97, 101] in MOOP. In the context of

MOOP in both optimizing query processing and DICOM data storage configuration in a cloud federation, a challenging problem is how to optimize a Multi-Objective Problem with a low computational complexity Multi-Objective Optimization algorithm.

In conclusion, in the context of the variable environment, MOOPs in both find query processing and data storage, and heterogeneous database environment in a cloud federation, challenging problems are how to optimize query processing and data storage of medical data in a cloud federation.

1.2 Related work

In a cloud federation, the solutions in [83, 109] provide various tools to manage heterogeneous data with multiple database engines. Authors in [83] proposed a functional SQL-like language that is capable of querying multiple heterogeneous data stores and optimizes query performance by minimizing data transfers, but it does not reduce the query execution time, monetary or other cost metrics. Besides, [109] showed a method to process a query which integrates data in PostgreSQL and MongoDB. It used the syntax and semantics of SQL++, which is unifying semi-structured data model and query language that is designed to encompass the data model and provide the capabilities of NoSQL, New SQL, etc. In their paper, they stated not to discuss cost optimization. While [144] focused on MOQP in a homogeneous database, Intelligent Resource Scheduler (IReS) [42] solved MOQP in heterogeneous systems by a specified policy which combined multiple objectives to a scalar value, not optimizing MOQP by building a Pareto plan set as other Multi-objective query optimization algorithms [144, 143]. This approach may significantly change the problem of nature. In some cases, the problem becomes harder to solve, or certain optimal solutions are not found anymore [60].

In addition, cost modeling solutions may be quite complicated with the variability characteristic. In the first class of cost modeling solutions, cost models introduced to build an optimal group of queries [103] are limited to MapReduce [36]. Besides, PostgreSQL cost model [153] aims to predict query execution time for this specific relational Data Base Management system. Moreover, OptEx [120] provides estimated job completion times for Spark [135] with respect to the size of the input dataset, the number of iterations, the number of nodes composing the underlying cloud. However, these papers only mention the estimation of execution time for a job, not for other metrics, such as monetary cost. Meanwhile, various machine learning techniques are applied

to estimate execution time in recent research [154, 141, 4, 55]. They predict the execution time by many machine learning algorithms. They treated the database system as a black box and tried 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.

Furthermore, to solve MOOP, the problems are often solved by turning the problem into a SOOP first and then solving that problem. However, Single-Objective Optimization Problems cannot adequately represent multi-objective problems [60]. Moreover, a large space of candidates leads to the necessity of finding a Pareto set of data storage configurations in MOOP. Besides, generating a Pareto-optimal front is often infeasible due to high complexity [160].

Meanwhile, Evolutionary Algorithms (EAs), an alternative to the Pareto-optimal, look for approximations (set of solutions close to the optimal front). For example, EMO approaches [72, 81, 161, 40, 39, 122] have been developed based on Pareto dominance techniques [72], Pareto Archived Evolution Strategy (PAES) [81], Strength Pareto Evolutionary Algorithm (SPEA) [161]. EMO often focused on both convergence and diversity [118]. In particular, Non-dominated Sorting Genetic Algorithms (NSGAs) [40, 37] aim to reduce the computational complexity while maintaining the diversity among solutions. NSGA-II [40] and SPEA-II [161] use crowding distances to maintain diversity. However, the computational complexity is still high, while the diversity cannot be preserved with more than two objectives [76]. Zhang and Li [158] proposed MOEA/D to maintain diversity with more than three objectives problem. MOEA/D uses a decomposition approach to divide multiple objectives into various single objective optimization sub-problems and solves up to four objectives [118]. Besides, Deb and Jain [39] proposed NSGA-III, which uses a set of reference directions to guide the search process. However, these algorithms still have a high computational complexity.

Moreover, recent works [63, 46, 97] have been proposed to optimize the hybrid data storage configuration. However, HYRISE [63] and SAP HANA [46] do not consider the high volume and sparsity of DICOM data. Although [97] provides an automatic approach producing data storage configurations for DICOM data, authors claimed that the space of candidate solutions in MOOP is large, but did not give any method to find the optimal hybrid data storage configurations. It needs a method to find an approximate optimal solution for DICOM data storage.

In conclusion, to the best of our knowledge, the existing solutions do not address all the problem of medical data management in cloud federation with Multi-Objective Optimization Problems (MOOPs) for (1) query processing and (2) data storage.

1.3 MIDAS

Medical data management in a cloud federation leverage a solution of MOOP in variability, heterogeneous cloud environment. The variability characteristic of a cloud federation raises an issue of accurate estimation. Besides, MOOPs in both of query processing and data storage optimization problem need an alternative solution to find a Pareto-optimal. As a consequence, all solutions of medical data management should be integrated into the heterogeneous database systems. Therefore, our motivation is to solve MOOPs of query processing and data storage configuration of a medical data management in a cloud federation.

First, we propose a Medical system on cloud federations (MIDAS). Our system is a cloud federation for medical data organizing and management. The system can manage the tremendous growth of medical data volume in the heterogeneous environment. Among various solutions of a heterogeneous system, IReS platform has advantages and could be extensible for more algorithms and tools. Hence, we extend IReS platform to manage and integrate our solutions in the various database engines environment. An overview of MIDAS is shown in Figure 1.1.

Second, we propose an algorithm for estimating of cost values in a cloud environment, called Dynamic REgression Algorithm (DREAM) to avoid entire historical data and inefficient processing for estimating cost values based on machine learning. This approach adapts the variability of cloud environment by changing the size of data for training and testing process to avoid using the expired information of the systems. Figure 1.2 shows the way of using historical data of DREAM in our system, where the entire historical data is Training set for DREAM and the new training set is the output of DREAM. This training set is used to build a cost model in Modelling module.

Third, we introduce Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) to solve the problem of high computation complexity NSGAs. MOOPs often have a large candidate space, and leverage an alternative approach, NSGAs, in finding an approximate optimal in a cloud federation is often used. However, the qualities of NSGAs need to be improved, and they still have high computation complexity.

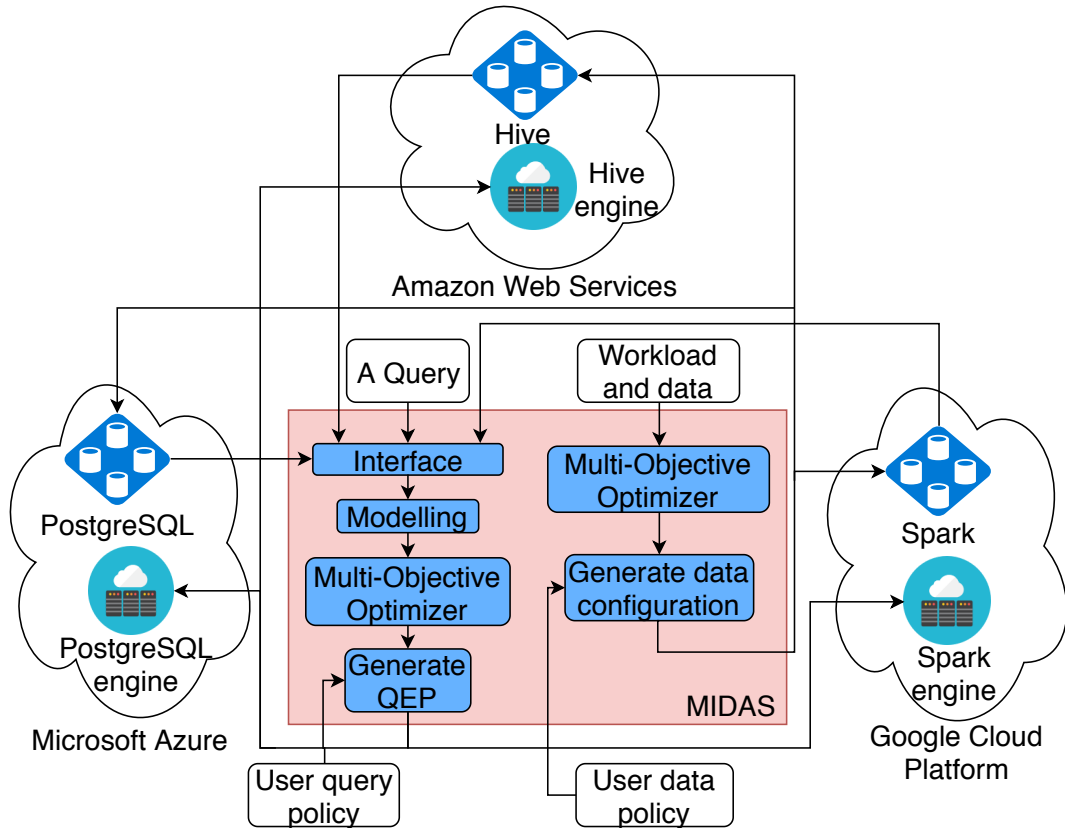


Figure 1.1 – An overview of MIDAS

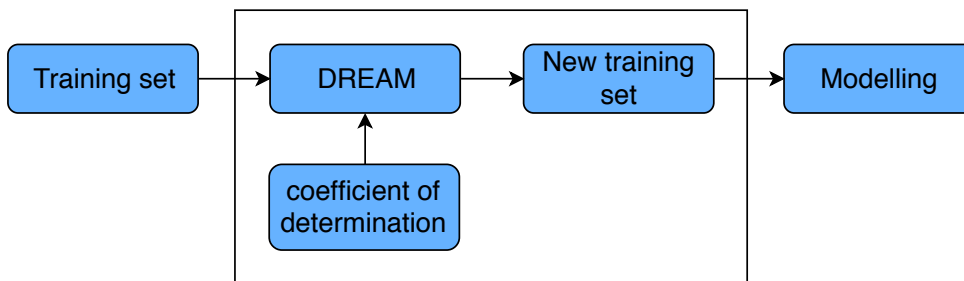


Figure 1.2 – Using DREAM to build cost models.

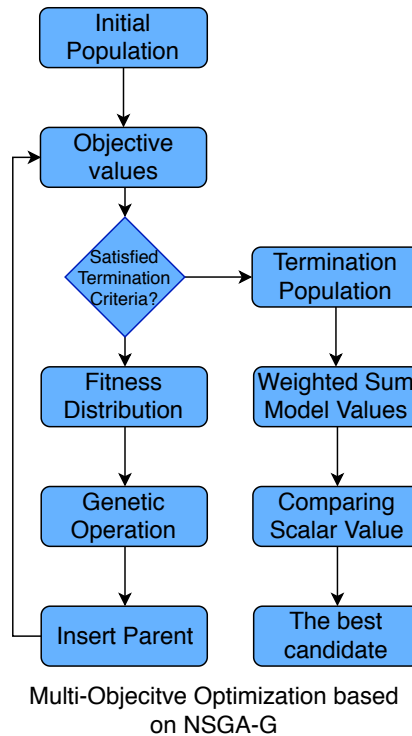


Figure 1.3 – An overview of Multi-Objective Optimization based on NSGA-G.

NSGA-G not only has low computation complexity but also improves the qualities of NSGAs. The algorithm is applied to Multi-Objective Optimizer in our system, as shown in Figure 1.3.

Finally, the algorithms above can be applied to MOOP in the cloud environment, including finding data storage configuration for DICOM in a cloud federation. We propose to use NSGA-G to find an approximate optimal solution for DICOM data storage configuration.

The contributions of this thesis are as follows,

- Introducing a medical system on a cloud federation called MIDAS. It is based on the Intelligent Resource Scheduler (IReS) [42], an open source platform for complex analytics workflows executed over multi-engine environments.
- Presenting Dynamic REgression AlgorithM (DREAM) to provide accurate estimation with the low computational cost. DREAM focuses on the increasing accuracy estimation and reducing computational cost in the variability of cloud federations.
- Presenting 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 Pareto-optimal for MOOP of DICOM hybrid store. NSGA-G maintains the diversity by randomly selecting solutions in a Pareto set in multiple groups, which are divided by a Grid Partitioning in the space of solutions. A solution is selected by comparing members in a group, instead of all members in a Pareto set. NSGA-G does not only inherit the superior characteristics of NSGAs in computational complexity but also improve both quality and computation time to solve MOOP.

- The project Java code and experimental data can be found at the following URL
 - + <https://gitlab.inria.fr/trle/IReS-Platform>. MIDAS is presented as an extension of IReS platform. MIDAS is validated in a private cloud, Galactica platform, with some database engines, such as Hive, PostgreSQL, Spark. Besides, DREAM and NSGA-G are also integrated in this project in both query processing and data storage configuration. DREAM is compared to the accuracy of estimating cost values to other machine learning algorithms. In the experiment, we use the TPC-H benchmark dataset and a variability environment to validate DREAM. Furthermore, DICOM dataset is also experimented by NSGA-G in the finding optimal solution test.
 - + <https://gitlab.inria.fr/trle/moea>. NSGA-G algorithm is integrated into the MOEA framework as an algorithm. The running example is also provided to compare qualities among multi-objective optimization algorithms. The qualities of NSGA-G is compared to other NSGAs with many problems in MOEA framework. In NSGA-G experiment, we use DTLZ and WFG problems in comparing the quality of various Multi-Objective Genetic algorithms.

1.4 Outline

This thesis is organized as follows.

- Chapter 2 introduces an overview of cloud federations.
- Chapter 3 presents an overview of medical data management on clouds.
- Chapter 4 shows the MOOP and NSGAs solutions.
- In Chapter 5, we introduce an algorithm of accurate estimation with low computation cost for the variability environment of clouds.
- A Non-dominated Sorting Genetic Algorithm based on Grid Partitioning is pre-

sented Chapter 6.

- An approach of finding an optimal data storage configuration for a medical data system is introduced in Chapter 7.
- Experiments of DREAM and NSGA-G are shown in Chapter 8.
- The conclusion, future work is presented in Chapter 9.

PART I

State of the art about cloud federation

CLOUD FEDERATIONS

Contents

2.1 Introduction	12
2.2 Definitions	13
2.2.1 Motivating Example	13
2.2.2 Big data Management System	15
2.2.3 Cloud Computing	17
2.2.4 Cloud Federation	18
2.3 Resource Management	19
2.3.1 Virtualization	19
2.3.2 Partitioning	20
2.4 Data Management	21
2.4.1 Homogeneous system	21
2.4.2 Heterogeneous systems	22
2.5 Conclusion	25

2.1 Introduction

The main goal of this chapter is to introduce the background and the existing solutions in clouds and cloud federations. First, we give a motivating example and the definitions in Section 2.2. Then, we discuss the different ways of resource management in Section 2.3. Finally, data management approaches are represented in Section 2.4.

2.2 Definitions

This section shows the motivating example and the definition and related techniques. First, medical data has the 3Vs characteristic of big data, such as high volume, high variety, and high velocity. Hence, medical data should use big data management system to organize a huge set of medical data. Second, cloud computing technology enables to build and access resources which helps processing and storing data in a public cloud, instead of local servers. Third, we present a concept of interconnecting the cloud environments of more than one service providers for multiple purposes of commercial, service quality, and user's requirements which is called cloud federation.

2.2.1 Motivating Example

In general, cloud federation may lead to query data across different clouds. 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: high volume, high variety, and high velocity. The data stores that belong in different clouds are shown in Figure 2.1. This example shows that the data can be stored in three different clouds, such as Amazon Web Services, Microsoft Azure, Google Cloud Platform. Each of them can be managed by different database engines, i.e., Hive, PostgreSQL. Hence, a cloud federation is thus a critical issue in terms of multi-engine environment.

Cloud computing [5, 8] allows resources (e.g., CPU and storage) to be provided as a service through the Internet by users' demand. According to the pay-as-you-go 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 2.1 shows the pricing of instances in two cloud providers. The price of Amazon instances is lower than the price of Microsoft instances, but the price of Amazon is without storage. Of course, this is only a small example of a pricing scheme in Amazon and Microsoft cloud service. There are many other pricing schemes depending on the kinds of instance, storage, location of servers. Hence, depending on the demand of a query, the monetary cost is

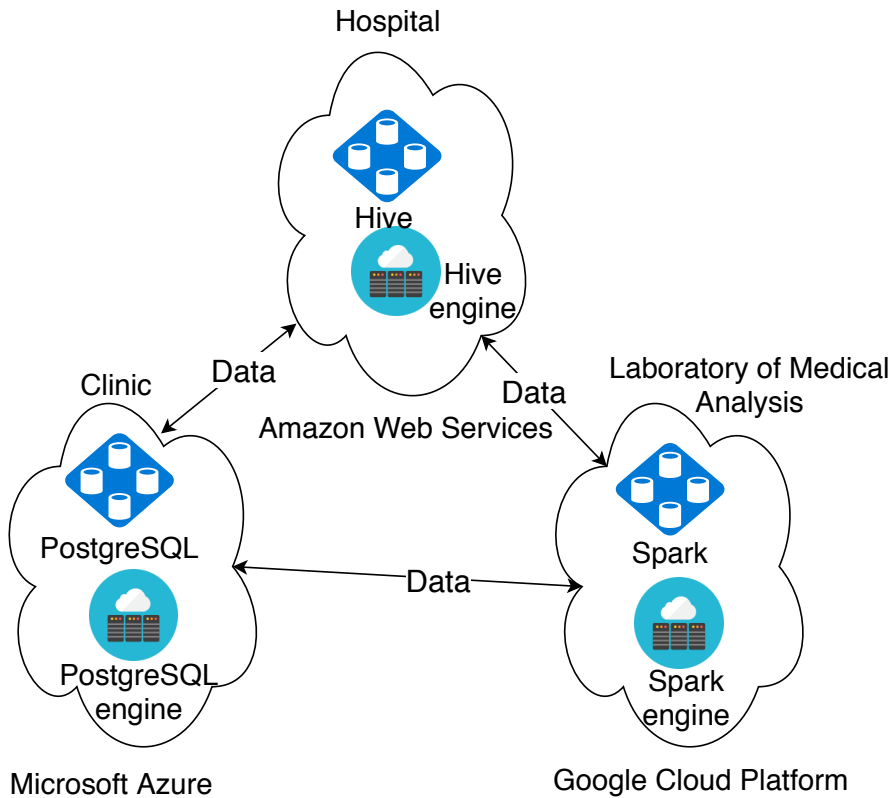


Figure 2.1 – Motivating Example on using cloud federation.

lower or higher at a specific provider. Besides, the response time of query processing is different in various instance configurations. The objectives in a cloud federation may be contradictory. For instance, the monetary cost is proportional to the execution time in the same virtual machine configuration in a cloud. However, the power virtual machine has high pricing will process a given query in a shorter time than the weak one which has low pricing. Hence, the execution time and the monetary cost are contradicting objectives. For example, there is a query Q in the medical domain. This query is optimized and transformed into a Query Execution Plan (QEP) which respects the infrastructure and its configuration. A various QEPs are generated with respect to the number of nodes, their capacity in terms of CPU, memory and disk and the pricing model. Table 2.2 presents an example of possible QEPs for Q . Choosing an execution plan is a trade-off between the minimization of the response time and the monetary cost. As a consequence, a cloud federation is thus a critical issue in terms of Multi-Objective Optimization Problem.

Table 2.1 – Example of instances pricing.

Provider	Machine	vCPU	Memory (GiB)	Storage (GiB)	Price
Amazon	a1.medium	1	2	EBS-Only	\$0.0049/hour
	a1.large	2	4	EBS-Only	\$0.0098/hour
	a1.xlarge	4	8	EBS-Only	\$0.0197/hour
	a1.2xlarge	8	16	EBS-Only	\$0.0394/hour
	a1.4xlarge	16	32	EBS-Only	\$0.0788/hour
Microsoft	B1S	1	1	2	\$0.011/hour
	B1MS	1	2	4	\$0.021/hour
	B2S	2	4	8	\$0.042/hour
	B2MS	2	8	16	\$0.084/hour
	B4MS	4	16	32	\$0.166/hour
	B8MS	8	32	64	\$0.333/hour

Table 2.2 – Multiple Objectives for Query Execution Plans

QEP	Vms	Price (\$/60min)	Time (min)	Monetary (\$)
QEP1	10	0.02	60	0.2
QEP2	40	0.02	22	0.29
QEP3	30	0.02	26	0.26

2.2.2 Big data Management System

As the motivating example in the previous section mentioned that the 3Vs characteristics of medical data are leverage using big data management systems. In this section, we describe the database technology which addresses the big data management system.

There are two main classifications of database technology, relational databases, and non-relational databases. Relational databases uses structured data and the structured query language, SQL, while non-relational databases implement document-oriented and non-structured query languages. For example, Amazon Relational Database Service (Amazon RDS) provides various relational database engines, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server, and non-relational databases, such as Amazon DynamoDB, Amazon DocumentDB.

A database management system often use effectively structured database. These databases are widely used and supported. Some examples of relational databases are the following:

- MySQL. A popular open-source Structured Query Language (SQL) database.

- Oracle. A C++ based object-relational Database Management System (DBMS).
- IBM DB2. An IBM big data analytics product.
- Sybase. A relational database, the first enterprise-level DBMS for Linux.
- MS SQL Server. A relational database is developed by Microsoft. It supports both SQL and Non Structured Query Language (NoSQL) database.
- MariaDB. A version of MySQL.
- PostgreSQL. Another object-relational.

However, the massive volume of data which has grown too big to be managed and analyzed by traditional data processing tools [104]. The relational database management systems were developed for a long time ago when the hardware, the storage, and resources were very different than they are today [124]. The relational databases are developed for structured data, so they do not work well with unstructured data. Non-relational databases, called NoSQL, are generated to store, manage and analyze this unstructured data. While Relational DataBase Management System (RDBMS) might not scale out easily on commodity clusters, NoSQL databases expand easily to take advantage of new nodes. Non-structured data includes many kinds of data, such as text, images, social media data, video, etc. They cannot be organized in tables like structured data. Some kind of NoSQL are:

- Key-value model. The model use indexed keys and values to store data. For example, LevelDB [89], and Riak [114].
- Column store. The tables are stored in columns. This method allows the system has better scalability and performance. For example, Apache Cassandra [7], BigTable [16], HBase [64], HyperTable [70].
- Document database. Unique keys and data are used to present documents. For example, CouchDB [32], MongoDB [96].
- Graph database. The data connections are presented by a graph. For example, Neo4J [100].

Large scale data processing

Additional, there are many frameworks supporting large scale data analytics [149]. In particular, MapReduce [152], Spark [135] and Hadoop [128] are popular open source frameworks for large scale data analytics [119]. Various researches efforts investigate to improve performance of data management in the cloud, i.e., Spark SQL [10], Hive [139, 140].

Spark SQL [10] is a module in Spark. It is integrated relational processing with the functional programming Application Programming Interface (API) of Spark. Spark SQL includes a DataFrame API for relational operations on both external data sources and built-in distributed collections and Catalyst, an extensible optimizer. The Catalyst Optimizer is designed as a new extensible optimizer based on functional programming constructs in Scala. Catalyst provides rule-based and cost-based optimization. The optimization process goes through four phases: (1) analyzing a logical plan, (2) logical plan optimization, (3) physical planning, and (4) code generation. In the physical planning phase, multiple plans are generated and compared based on cost. The rule-based is used in other phases. However, the cost-based optimizer is built on a single objective optimization problem, and the plan cost is depended on the prefix cluster configuration before the optimization process.

Hive [139, 140] is an open source data warehousing solution built on top of Hadoop. In this open source, Hive process queries in a SQL-like declarative language, which is compiled into MapReduce jobs executed on Hadoop. Hive stores data in tables as the traditional databases with different formats. Hive support different file formats such as TextFile, SequenceFile, RCFile, AVRO, ORC, and Parquet. Hive is designed for the scalable analysis on large data sets. They are enhancing the Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) drivers for Hive for integration with commercial BI tools.

In conclusion, the above methods provide different data managements in the cloud environment. However, various hospitals maybe use different systems. As a consequence, medical data management in the cloud environment needs to consider heterogeneous systems.

2.2.3 Cloud Computing

While big data management systems represent techniques to organize huge sets of data, cloud computing provides the platform to share computer facilities. In this section, we describe the cloud computing technology which is used to process, store data, in a network of remote servers.

Cloud computing appeared in Compaq internal document [1] in 1996. Till 2006, the term “cloud computing” was popularized with Amazon [6]. Cloud computing [5, 8, 9] is a large-scale model for demanding resources as a service over the Internet, such

as computing, storage, networks, platforms, and applications. The evolution of cloud providers in virtualization, the high-speed Internet raises the development of cloud computing, one of the fastest growing fields in IT industry [5]. The most attractive properties of cloud computing are efficiency and flexibility which enables to build and access resources without the installation and management. The users can pay and have the system at any time on a pay-as-go-go model.

To get full benefits of cloud computing, users should know the advantages and limitations of this technology. The main aspects of cloud computing are presented as follows:

- On-demand self-service. The cloud providers provide on-demand self-service to automatically provision cloud resources on demand whenever users require them. The users can access cloud services through a web-based management interface.
- Broad network access. The users can remotely access cloud resources over the network with heterogeneous client platform such as different operating systems, mobile devices, workstations.
- Resource pooling. The resource pooling is built by cloud providers to serve customers using techniques based on virtualization technologies. The resources are dynamically used and reused by consumers without knowledge about physical resources.
- Measured service. The cloud providers present multiple services to monitor, control, account and transparent resources to enable using the pay-as-you-go model.

2.2.4 Cloud Federation

As defined in [29], cloud federation is a concept of service aggregation characterized by interoperability features. This definition addresses the economic problems of vendor lock-in and provider integration. Besides, the Inter-Cloud is defined by the Global Inter-Cloud Technology Forum (GICTF) [51]: “A cloud model that, for the purpose of guaranteeing service quality, such as the performance and availability of each service, allows on-demand reassignment of resources and transfer of workload through a interworking of cloud systems of different cloud providers based on coordination of each consumer's requirements for service quality with each provider's SLA and use of

standard interfaces". This thesis suggests a definition of the term Cloud Federation, a concept of interconnecting the cloud environments of more than one service providers for multiple purposes of commercial, service quality, and user's requirements.

Based on our context, there are various aspects of a cloud federation, including resource management, data management in the heterogeneous environment. The next sections present the state of the art of these aspects, such as resource management and data management in the cloud federation.

2.3 Resource Management

Resource management is the process of organizing resources: planning, scheduling, allocating. Thus this is of major interest for users to find a good solution for their business.

2.3.1 Virtualization

Virtualization is a solution to replace some physical component with Virtual Machines (VMs). This technology allows to reduce the amount of hardware in use. Cloud computing technology uses the pay-as-you-go model to leverage virtualization and provide on-demand resource provisioning over the Internet [13]. Users can decrease the costs of maintenance of their hardware on the computing environment.

Minimizing resource utilization when deploying query plans on multi-core machines is studied in [59]. They propose an algorithm to optimally assign relational operators to physical cores. The system is built on top of SharedDB [58] and can be applied to other shared work systems. The method focuses on physical resources, such as CPU, memory. Hence, the system is suitable for a specific shared work system.

Depending on the workload, AGILE [102] uses dynamic Virtual Machine cloning to reduce application startup times. It also predicts demands to provide a medium-term resource. Among different prediction algorithms, they use a simple Markov model to minimize the prediction error. However, the model predicts the demanding CPU only. The configuration of Virtual Machine should consider other resources such as memory, network bandwidth and disk I/O.

ElasTraS [34] provides an elastic, scalable and self-managing transactional database tool for the cloud. They consider Online transaction processing (OLTP) queries. The

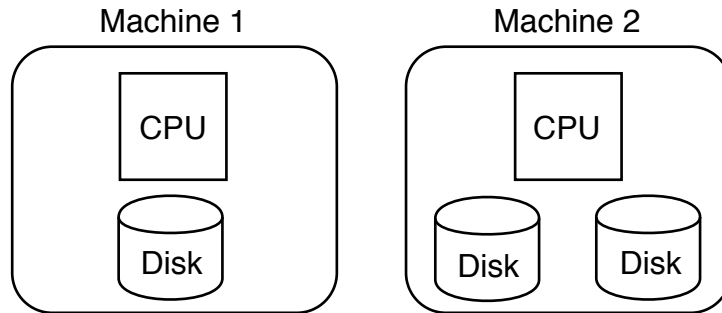


Figure 2.2 – A simple cluster.

system uses many metrics, including service unavailability, number of failed requests, impact on response time, and data transfer overhead. However, they do not propose an algorithm for MOOPs. Besides, authors state that the system should improve prediction accuracy, better workload consolidation, and partition assignments.

In conclusion, recent resource management publications show the way to optimize the virtual machine configuration in the cloud environment. However, users consider more than one metric, such as the execution time, the monetary cost, etc. Hence, the resource management should consider MOOP in the cloud federation.

2.3.2 Partitioning

Partitioning is a strategy to enable load balancing on several nodes. In a heterogeneous cluster, the uniform data partitioning method may overload some weak machines and underutilize the strong machines. For example, a cluster consisting of two machines is illustrated by Figure 2.2. The disks of the first machine are 50% slower than that of the second machine. Suppose that a query needs to scan a table and count the number of tuples in the table. The query completes when both machines finish their processing. If 50 % data are assigned to each machine, the second one will finish the job before the first machine does. To minimize the total execution time, it is easy to assign the best partitioning scheme with 33% of data to the first machine and 67% of data to the second machine. As a consequence, the workload has similar response times in both machines.

For a given workload in heterogeneous environments, [90] is developed to quantify performance differences among machines. They use the static and dynamic data partitioning strategy to improve workloads on heterogeneous clusters. In the optimization

problem, they try to minimize execution time with a given budget by selecting the most suitable computing resources for building a cluster. However, in the cloud federation, we should consider the transfer data between clouds because the price of data in/out is different in various cloud providers. Two machines may be in two clouds with different pricing scheme. Hence, the monetary cost of the second scheme is higher than the first one.

2.4 Data Management

Data management is a process including acquiring, validating, storing, protecting, and processing data. Data management encompasses various techniques. This section classifies them into two classes: homogeneous and heterogeneous.

2.4.1 Homogeneous system

In homogeneous database systems, all distributed clusters have identical data management systems. Following researches studied the homogeneous database system.

A stochastic metaheuristics [20] method is considered to the problem of data distribution in the cluster. Besides, a data redistribution for optimization of load-balancing quality is also considered in [20]. The experiments show that the database control system can use the load-balancing quality metrics and stochastic meta-heuristics method to optimize data management in clouds. This system does not concern the heterogeneous database system in a cloud federation.

Lookahead Information Passing (LIP) is introduced to collapse the space of left-deep query plans for star schema warehouse down to a single point near the optimal plan [159]. To avoid a poor join order from the optimizer, they pass succinct filter data structures from the outer relations in all the joins to the inner relation. They implement the LIP technique in the RDBMS.

An efficient multidimensional subspace skyline computation [87] uses point-based space partitioning to improve the execution time. They compare their proposed algorithm to the existing skycube algorithms. The experiment shows that the proposed algorithms are significantly faster than the state-of-the-art algorithms.

An effective parallel algorithm for probabilistic skyline queries is introduced in [110]. The uncertain data models including discrete and continuous are considered.

Three papers show the way to generate query plans in the left-deep or skyline methods. All of them focus on a specific system. The following section describes more complex systems in designing and managing, heterogeneous database system.

2.4.2 Heterogeneous systems

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. This section presents the federated database engines and the limitation in cloud federations.

Proteus

Supporting heterogeneous data formats and minimizing query execution times are also the concern of Proteus [73]. Proteus is a query engine which supports queries over CSV, JSON, and relational binary data. Proteus adapts data caches to the workload trends. However, Proteus optimizes query on execution times, and focuses on the mono objective problem.

Polystore Query Rewriting

Polystore Query Rewriting [109] designs a mediator, i.e., FORWARD Middleware to evaluate queries over different databases. This model is designed to encompass the data model and query language capabilities of NoSQL, and SQL. Figure 2.3 show an example of FORWARD query processing. To integrate data stored in NoSQL database and SQL database, a query is rewritten into a plan, which includes PostgreSQL and MongoDB subqueries. The subqueries are processed in the appropriate database engines. However, they stated that they do not discuss cost optimization and Polystore design. The sub-queries are optimized on a specific database based on its own optimize with a single objective problem. Hence, this model is not suitable for cloud federation.

BigDAWG Polystore System

The Big Data Analytics Working Group (BigDAWG) [45] project is built on federated databases over multiple data models [52, 91], specialized storage engines [123], and

visualizations for Big Data [98]. Figure 2.4 shows the architecture of BigDAWG [45] project. They call their architecture a polystore to distinguish it from previous federation efforts that used only the relational model. To enable users to enjoy the performance advantages of multiple vertically-integrated systems, they implement island of information, including shim for interacting. BigDAWG uses CAST operators to move data between engines. However, it is developed based on integrating multiple database engines. Hence, they optimize a query by the engine database optimizer, not MOOP in the cloud federation.

CloudMdsQL

Cloud Multidastore Query Language (CloudMdsQL) [82] [83] is a functional SQL-like language, which can exploit the full power of local data stores. The query optimizer in CloudMdsQL rewrite queries and optimize them by using cost functions or database statistics, simple cost model. Hence, they do not consider the Multi-Objective problems in the optimization process.

MuSQLE

Distributed SQL Query Execution Over Multiple Engine Environments (MuSQLE) [57] is a system for SQL-based analytics over multi-engine environments, as shown in Figure 2.5. The framework develops a novel API-based strategy. MuSQLE engine API is integrated with a state-of-the-art query optimizer, adding support for location-based, multi-engine query optimization and letting individual run-times perform sub-query physical optimization. The derived multi-engine plans are executed using the Spark distributed execution framework [135]. In particular, they integrate PostgreSQL [133], MemSQL [131] and SparkSQL [136] under MuSQLE and demonstrate its ability to accurately choose the most suitable engine. Although MuSQLE can process a query in a distributed environment and heterogeneous database system, it optimizes queries based on a single objective optimizer in each engine. Besides, the configuration of MuSQLE is predefined. Hence, the optimizer estimates the cost values and gives the optimization solution based on this constant configuration. In a cloud federation, a given job can generate many solutions which their metric values are depended on the different configurations.

MISO

MISO [86] is designed to reduce the amount of data movement during query processing. The big data analytic framework is integrated with traditional RDBMS to allow a query accessing data and computing in both row and column stores. However, they optimize a query on a single objective problem and do not consider the monetary cost in the pay-as-you-go model in the cloud environment.

Polybase

Polybase [35] is presented to allow managing and querying data stored in a Hadoop cluster using the standard SQL query language. Both structured data in a relational DBMS and unstructured data in Hadoop can be used to execute queries. Polybase optimizes queries on a cost-based query optimizer. It is a mono objective optimization problem. Besides, how to optimize storing data is not addressed in the system.

Estocada

Estocada [23] is an architecture for heterogeneous datasets based on different data stores. The query is optimized by local-as-view integration and view-based rewriting. However, authors do not consider the multiple objective problems and the variability of the cloud environment.

IReS

IReS [42] is an open source platform for managing, executing and monitoring complex analytics workflows. IReS provides an optimizing cost-based workflows method. A customizable resource management of diverse execution and various storage engines are also presented. **Interface** is the first module which is designed to receive information on data and operators, as shown in Figure 2.6. **Modelling** module is used to predict the execution time by a model chosen by comparing machine learning algorithms. For example, Least squared regression [115], Bagging predictors [22], Multilayer Perceptron in WEKA framework [138] are used to build the cost model in **Modelling** module. **Model DB** optimizes a work flow based in the cost model. **Planner** generates an execution plan. Finally, the job is run on multiple engines, as shown in Figure 2.6.

Table 2.3 – Advantage and disadvantage of recent researches.

Research	Heterogeneous	MOOP
Proteus	✓	×
Polystore Query rewriting	✓	×
BigDAWG Polystore System	✓	×
CloudMdsQL	✓	×
MuSQLE	✓	×
MISO	✓	×
Polybase	✓	×
Estoscada	✓	×
IReS	✓	✓

In conclusion, the advantage and disadvantage of heterogeneous database systems are described in Table 2.3. As can be seen in Table 2.3, only IReS platform considers both heterogeneous systems and MOOP in clouds. The optimizer in IReS uses machine learning approaches to estimate the cost values of candidates of workflow based on the historic data and the configuration of clusters. However, they often use all of the historic data to train and build the model. It may lead to use expired information. Hence, this thesis aims to integrate our proposal into IReS platform to improve the accuracy of estimated values, while limiting computational cost.

2.5 Conclusion

This chapter presents works on two main aspects of cloud computing, i.e., resource and data management. Most of them do not consider MOOPs. Only IReS platform considers the problems of cloud environment, but expired information in historic data may affect the estimation cost values during the optimization process. Hence, the thesis proposes an algorithm to improve the accuracy of the estimation process. Also, to solve the heterogeneous problem, the thesis reuses the open source platform, IReS, to evaluate the proposed approach. Besides, another aspect of the cloud environment is the data storage configuration. How to configure data also affects the quality of the query processing. The next chapter will address optimization in general, and the case of medical data storage configuration in particular.

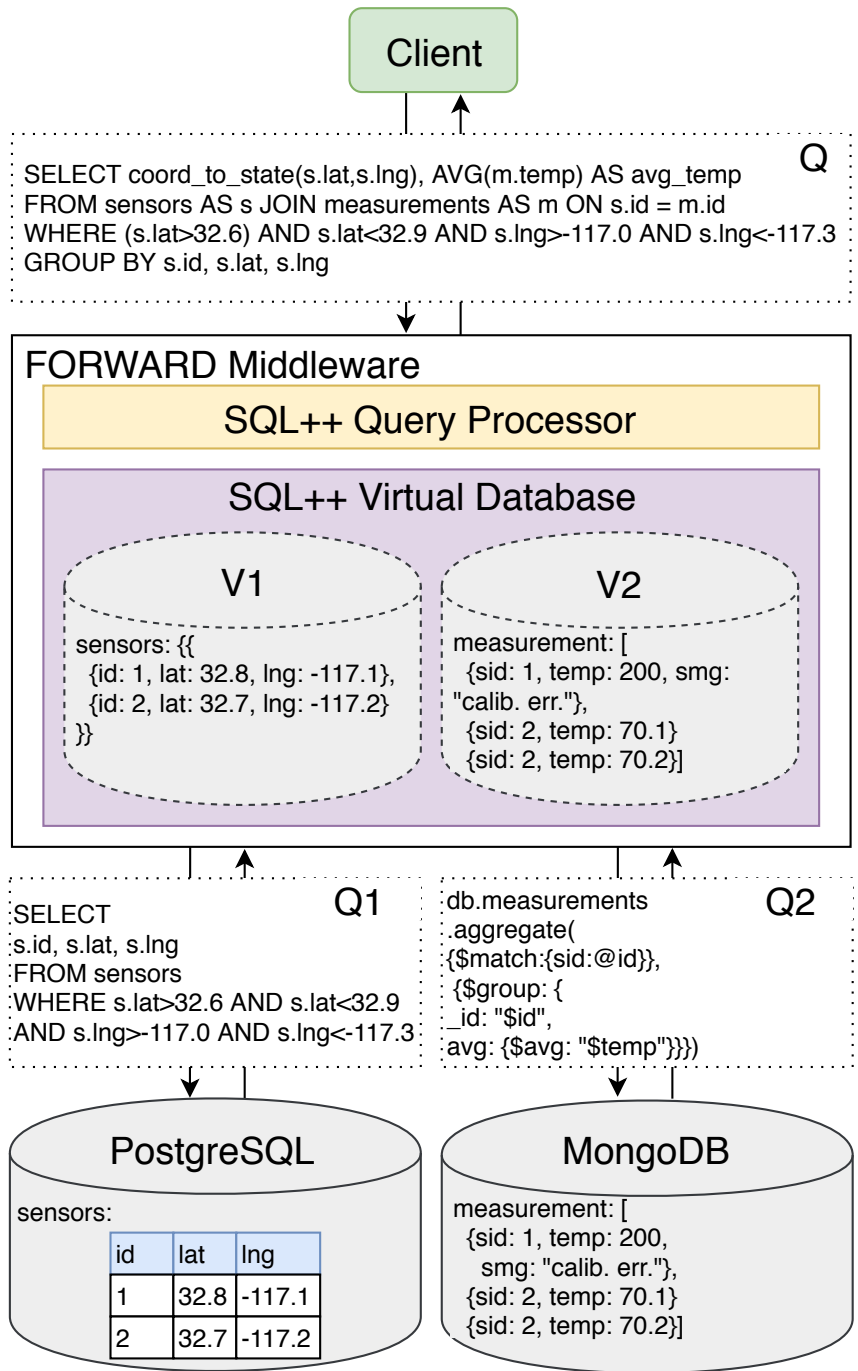


Figure 2.3 – FORWARD Query Processing [109].

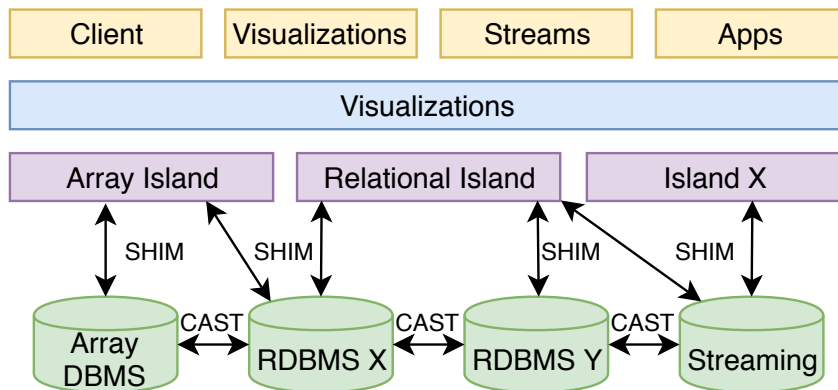


Figure 2.4 – BigDAWG Architecture [45].

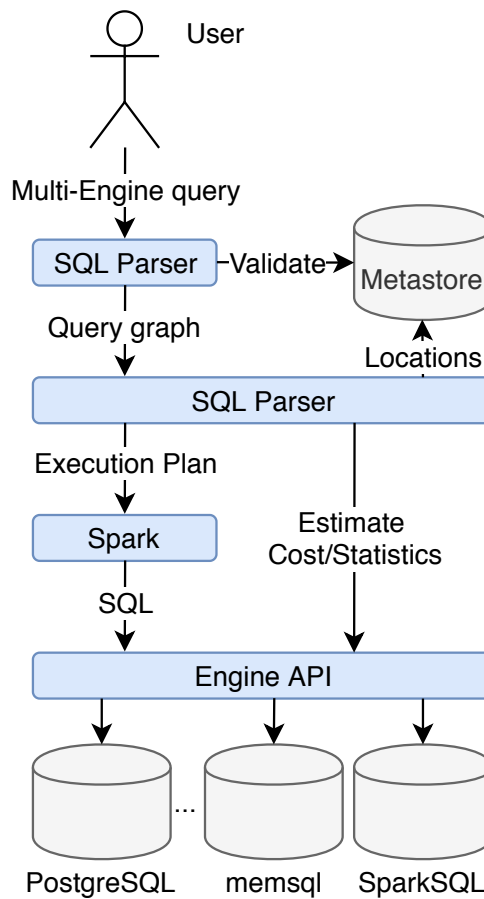


Figure 2.5 – MuSQLLE system architecture [57].

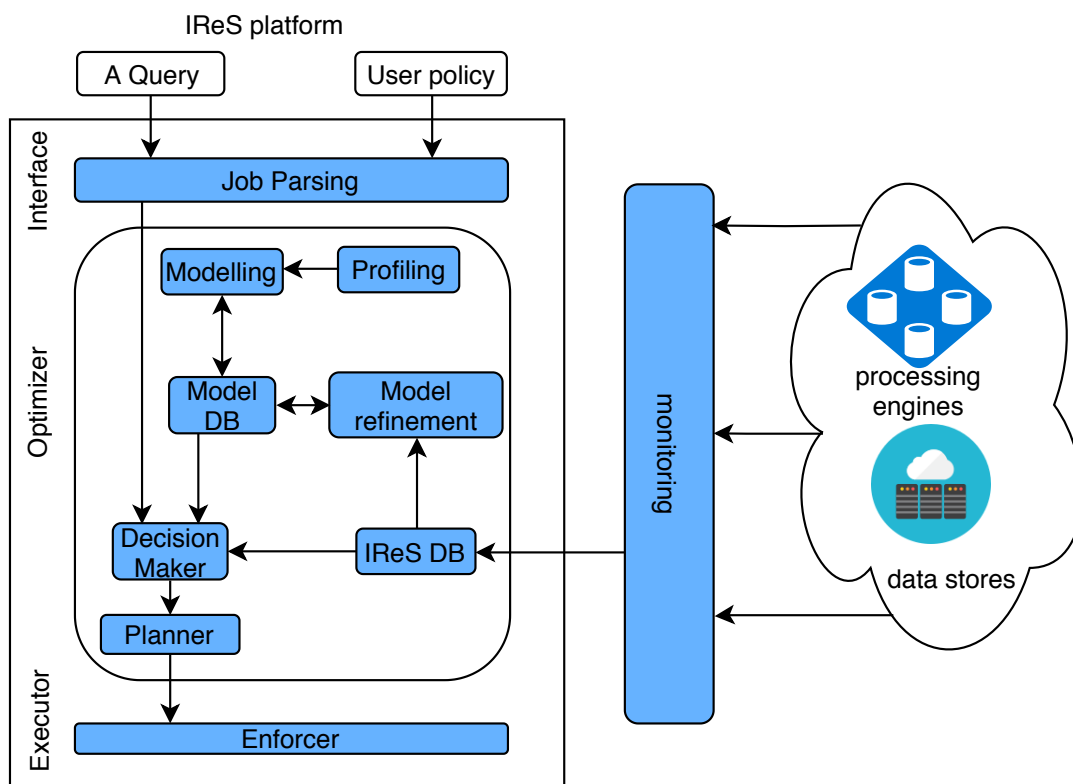


Figure 2.6 – Architecture of IReS [42].

OPTIMIZATION OF MEDICAL DATA MANAGEMENT

Contents

3.1 Introduction	29
3.2 Medical Data management	30
3.2.1 DICOM	30
3.2.2 Data Model	32
3.2.3 Hybrid data storage configuration	34
3.2.4 Vertical Partitioning	36
3.3 Search and Optimization	37
3.3.1 Single Objective Optimization	41
3.3.2 Multiple Objective Optimization	44
3.4 Conclusion	45

3.1 Introduction

Chapter 2 introduces the background and existing solutions in clouds and cloud federations. This chapter focuses on a specific use case, that is to say healthcare. It discusses opportunities and challenges in terms of data storage and possible layout configurations. We first describe the DICOM standard and then present medical data management using this format in Section 3.2. Also, hybrid data storage configuration approaches to optimize data storage are introduced. Next, we present the searching and optimizing approaches in Section 3.3. Finally, we conclude with Section 3.4.

3.2 Medical Data management

This section describes an overview of DICOM data and then presents the background of data storage configuration based on the data storage and query processing strategy.

3.2.1 DICOM

The international standard of medical data, DICOM [33], to transfer, store and display medical imaging information was first released in 1980 to allow interoperability among different manufacturers. DICOM applications are constructed and encoded the Data Set information following the Information Objects and Services Classes.

Information Object Definitions

DICOM Standard specifies the set of Information Object Definitions (IODs). A number of Information Object Classes provides the abstract definition of entities applicable to the communication of digital medical images and related information, such as waveforms, structured reports, radiation therapy dose, etc., in DICOM standard.

Service Class Specifications

A number of Service Classes is also defined in the DICOM standard. Information Objects are associated with Service Classes. For example, some Service Classes are shown as follows:

- Storage Service Class
- Query/Retrieve Service Class
- Basic Worklist Management Service Class
- Print Management Service Class.

Data Structure and Semantics

Various standard image compression techniques are supported in DICOM standard, e.g., JPEG lossless and lossy. The dataset containing the DICOM files in the white paper by Oracle [105] is created by six different digital imaging modalities. Its

Table 3.1 – Example of real DICOM data set [97].

Datasets	DICOM files	AttributesTuples	Metadata	Total size
CTColonography	98,737	86	7.76 GB	48.6 GB
Dclunie	541	86	86.0 MB	45.7 GB
Idoimaging	1,111	86	53.9 MB	369 MB
LungCancer	174,316	86	1.17 GB	76.0 GB
CIAD	3,763,894	86	61.5 GB	1.61 TB

Table 3.2 – Example of extracted DICOM data set [97].

Table	Number of Tuples	Size
Patient	120,306	20.788 MB
Study	120,306	19.183 MB
GeneralInfoTable	16,226,762	4,845,042 MB
SequenceAttributes	4,149,395	389.433 MB

total size is about 2 terabytes, including 2.4 million images of 20,080 studies. In particular, DICOM text files are used in [97], as shown in Table 3.2. They are extracted from real DICOM dataset, as shown in Table 3.1.

Characteristics of DICOM Data and workloads

Besides of the definitions above, DICOM specifies other standards, such as Data Dictionary, Message Exchange, Network Communication Support For Message Exchange, etc.

Some characteristics of DICOM [97] data are presented as follows:

- High Complexity,
- High Variety,
- High and Ever-increasing Volume,
- High Velocity.

The workloads in DBMSs can be divided into OLTP and Online analytical processing (OLAP). In general, OLTP systems focus on optimizing write operations, such as insert, delete, update, etc. The aim of OLTP systems is the very fast query processing. Row-oriented databases are often designed for OLTP applications. While, OLAP is characterized by a relatively low volume of transactions. OLAP systems focus on optimizing read operations, such as analyzing data. The aims of OLAP system is the response time. Column-oriented databases are often designed for OLAP applications.

Table 3.3 – Frequency of Queries in Workload W.

Queries	Detail	a1	a2	a3	a4
Q_1	SELECT * FROM T	1	1	1	1
Q_2	SELECT a1, count(a4) FROM T GROUP BY a1	1	0	0	1
Q_3	SELECT UID, a3 FROM T WHERE a3 = 'Modality'	0	0	1	0
Q_4	SELECT UID, a2 FROM T WHERE a2 = 'DA'	0	1	0	0

Having characteristics of the Big Data paradigm, DICOM data has been accessed by various OLTP, OLAP and mixed workloads. For example, table T , including 4 attributes: a_1, a_2, a_3, a_4 , is often accessed by a workload W as Table 3.3 shown.

As a consequence, how to design data storage for DICOM data is an issue for the medical system. The next section describes the different approaches in data storage configuration for OLTP, OLAP, and mixed workloads.

3.2.2 Data Model

For data management in general, and medical data in particular, there are two kinds of traditional storage for data model, row, and column data. Row stores store all attribute values in a row together. This kind of traditional data storage supports adding/modifying a row easily. It also helps efficiently reading many columns of a single row at the same moment. This strategy is suitable for OLTP workload, but wastes I/O costs for a query which requires few attributes of a table [66]. In contrast, column stores (e.g. MonetDB [19] and C-Store [125]) organize data by column. A column contains data for a single attribute of a tuple and stores sequentially on disk. The column stores allow to read only relevant attributes and efficiently aggregating over many rows, but only for a few attributes. Although the column stores are suitable for read-intensive (OLAP) workloads, their tuple reconstruction cost in OLTP workloads is higher than row stores.

Following sections show the strategies to optimize the data storage configuration. It is defined as a set of pairs consisting of (1) a group of attributes and (2) an associated data layout.

Data Storage Strategy

The first strategy in [97] aims to optimize query performance and storage space over a mixed OLTP and OLAP workload by extracting from the original DICOM files,

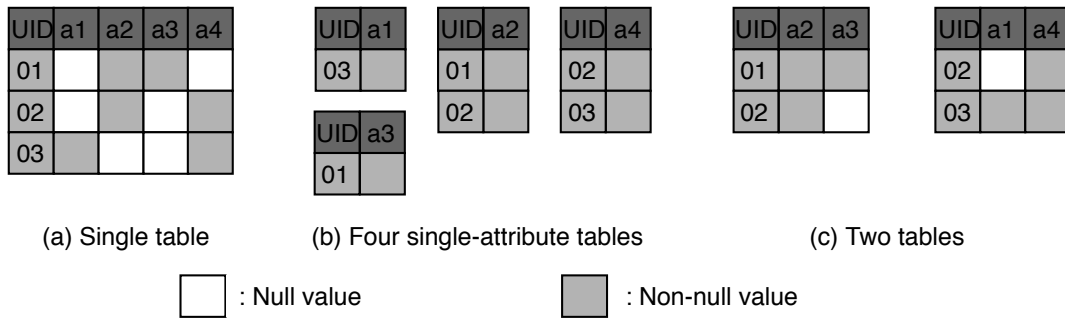


Figure 3.1 – Difference configurations of the table T .

organizing and storing data in a manner to reduce space, tuple construction and I/O cost. The data are organized into entity tables. The tables are decomposed into multiple sub-tables, which are stored in row or column stores of the hybrid store. A group of attributes classified as frequently-accessed-together attributes can be stored in a row table. Other groups are classified as optional attributes and stored in a column store. Each attribute belongs to one group except that it is used to join the tables together. This strategy removes the null rows in tables. For example, a given table T has four attributes as Figure 3.1(a) shown. Figure 3.1 presents three different data storage configurations, H_1, H_2, H_3 , of the table T : (a) the entire T is stored in a single a row store, $H_1 = \{\{UID, a1, a2, a3, a4\}, \text{row store}\}$; (b) the entire T is decomposed into 4 vertically partition tables, stored in a column store, $H_2 = \{\{\{UID, a1\}, \text{column store}\}, \{\{UID, a2\}, \text{column store}\}, \{\{UID, a3\}, \text{column store}\}, \{\{UID, a4\}, \text{column store}\}\}$; (c) the entire T is decomposed into 2 vertically partition tables, stored in a row store $H_3 = \{\{\{UID, a1, a4\}, \text{row store}\}, \{\{UID, a2, a3\}, \text{row store}\}\}$. If all the attributes of T are stored in a single row table, as Figure 3.1(a) shown, the number of data cells is 15. If table T is decomposed into four single-attribute tables, the number of data cells is 12. If T is divided into two tables, including attributed are often accessed together, Figure 3.1(c) shown, stored in row stores, the number of data cells is 12.

Query Processing Strategy

To improve performance of query processing in a distributed environment [97], the hybrid storage model of row and column stores needs to modify sub-tables to reduce the left-outer joins and irrelevant tuples in the input tables of join operations. The query performance is negatively impacted if the query execution needs attributes by joining

many tables. This query in hybrid stores [97] needs to reconstruct result tuples, and the storage space will increase to store surrogate attributes. For example, the table T in Figure 3.1 is often access by two queries, Q_1 and Q_2 . Where Q_1 often accesses to attributes a_2, a_3 , and a_1, a_4 are often accessed by Q_2 . Hence, the data storage configuration in Figure 3.1(c) is more recommended than other one in Figure 3.1(b).

In general, based on a given workload and data specific information, a large number of candidates of data storage configuration can be created for a given table. The number of candidates depends on the attributes, null values in tables, the number of database engines, etc.

3.2.3 Hybrid data storage configuration

To improve performances of storing and querying in OLAP, OLTP, and mixed workloads, DICOM data needs to be stored in a row-column store, called hybrid data storage. The next section presents previous works in hybrid data storage.

Column-Group Storage Models

Column-group storage models are built by organizing column groups in either row-oriented, column-oriented storage or both row/column-oriented storage to improve the performance of mix workloads.

In addition, the performance of modern databases are affected not only by the number of disk I/O operations but also by the main memory. The cost of main memory is decreasing, so some researches consider the modern database systems storing a large amount of data in main memory [27, 14]. However, the access latency between the processor and the main memory leakages another performance bottleneck. The modern database systems should use a cache memory between them to reduce that bottleneck [3]. If appropriate data is on the cache memory, the speed of processing data will increase, otherwise the processor should load the data from the main memory which reduces the speed of processing data. Hence, the cache often loads the frequently-used data to speed up data processing.

MemSQL

MemSQL [27] is a distributed SQL database. It is designed to exploit memory-optimized DBMS. In MemSQL, data can be stored in rows or columns. The first format is stored in a memory store and the second one is in a disk store. This model takes advantages of row, column data format on memory and disk data, respectively. It does not take advantage of the hybrid data format (row/column) on both memory and disk stores. MemSQL optimizes a query on a single objective problem (execution time).

HYRISE

HYRISE [63] is a main memory hybrid database system, as shown in Figure 3.2. The approach automatically partitions tables into vertical partitions of varying widths depending on how frequent columns in the original table are accessed in the original table. The mixed workload environment including OLAP and OLTP queries is analyzed to predict the layout performance on a hybrid row/column database. In particular, the tool prefers to suggest narrow partitions for OLAP queries and wide vertical partitions for OLTP queries. HYRISE uses an in-memory column-oriented database system. Frequently accessed together attributes are grouped in a vertical partition, called container allocated in main memory.

HYRISE provides an automated database design tool for given a schema, a query workload by using analytical model. However, the disadvantage of HYRISE is a main memory database system. It may store a small dataset depending on the size of the physical available memory. Hence, HYRISE maybe not suitable for a high demanding of ever-growing volumes of data such as DICOM data.

SAP HANA database

SAP HANA database [46, 47] is an in-memory row/column-oriented database system. Based on a same system, it optimizes data storage configuration depending on both OLAP and OLTP workloads with different characteristics of data, such as structured, semi and unstructured data. The system use multiple database engines. SAP HANA use the main memory to organize as much as possible based on frequently assessed data. The less frequently accessed data is stored in disks.

The advantage of SAP HANA is the size of data may be larger than the size of

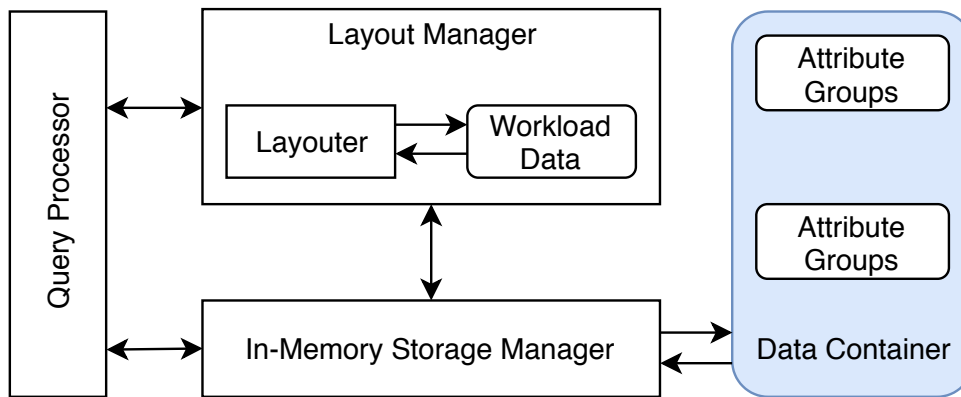


Figure 3.2 – HYRISE architecture [63].

physical memory. However, SAP HANA is limited to handle the high and ever-growing volume of data, such as DICOM data. Moreover, the disadvantage of it is lacking the automatic determining the row or column oriented data layout of a new table. It requires manually determine at definition time by the system administrator.

HYTORMO

HYTORMO is a hybrid layout relational system. It takes into account both storage and processing to optimize the layout of tables. The system uses both data storage and query processing strategy to optimize the layout of tables. In addition, the system applies Bloom filters [17] to reduce network I/O cost during query processing in cloud environment. HYTORMO is built on top of Spark and aims to be deployed on large scale environments.

An advantage of the tool is that it automatically designs the data storage configuration of hybrid data store based on four parameters following: the weight of similarity, clustering threshold, merging threshold and data layout threshold. However, the author states that the space of data storage configuration is too large. Hence, it is necessary to find an optimal data storage configuration in this data storage configuration space.

3.2.4 Vertical Partitioning

Vertical partitioning is a schema design technique to improve query processing performances, aiming to reduce I/O cost, and data storage. This technique is often used to reduce workload execution time and storage space size for sparse datasets. Verti-

cal partitioning algorithms are usually classified into two approaches: (1) cost-based or affinity-based and (2) top-down or bottom-up.

Cost-based vs. Affinity-based Approaches

Cost-based algorithms, such as [53], need an objective function to minimize the total workload execution cost of a given system. However, it is hard to build cost functions expressing complex execution mechanisms of query optimizers/engines of the current systems [108]. In contrast, affinity-based algorithms, such as [107], are based on attribute affinity (which shows how often attributes are simultaneously accessed by the same queries in a given workload) to cluster the attributes into clusters. However, the measured affinity is usually independent of the execution of the corresponding query optimizers or query engines of current systems. Hence, Thus, the target systems should further validate the result clusters [108]

Top-down vs. Bottom-up Approaches

Top-down algorithms, such as [2], usually decompose a schema containing all attributes into two smaller schemas. This decomposition is repeated similarly for each resulting schema until the given objective function cannot be further improved. In contrast to top-down algorithms, bottom-up algorithms, such as [65], begin with a set of minimally small vertical partitions. Each vertical partition may contain either a single attribute or a subset of attributes. For each step, a pair of vertical partitions are merged into a larger vertical partition. The step is repeated until the objective function cannot be further improved.

3.3 Search and Optimization

Managing medical data in a cloud federation requires to consider many metrics, such as, response time, monetary cost. It raises an issue of Multi-Objective Optimization Problem (MOOP). For example, MOOP compares query plans with various cost values, e.g., execution time, monetary cost. MOOP needs to optimize by Multi-Objective Optimization (MOO) approaches. The goal of MOO techniques is then to find the set of Pareto-optimal solutions, i.e. the query plans realizing optimal cost trade-offs

for a given query, the hybrid data storage configuration optimal cost trade-offs for a given workload and data set.

Besides, vertical partitioning approaches, including affinity-based algorithms [107], are widely used in a traditional databases. They use Attribute Usage Matrix and Frequencies matrices to optimize data in distributed database systems. HYTORMO [97] considers more about data specific information, a matrix containing the null values of the given table. This information did not appear in the traditional system.

In addition, most of recent hybrid data storage configuration provides the way to improve the performance of query processing, e.g., HYRISE, SAP HANA. However, they do not consider the high volume and sparsity of DICOM data (the null values). While HYTORMO consider the high volume and sparsity of DICOM data and mixed OLTP/OLAP workloads in the automatic generating hybrid data storage configuration. A problem with HYTORMO is that is does not provide the best solution in the space of parameters, such as data storage strategy ($\alpha, \beta \in [0, 1]$) and query processing strategy ($\theta, \lambda \in [0, 1]$). It needs to search and optimize the hybrid data storage configuration based on Multi-Objective Problem (MOP).

Both of hybrid data storage configuration and query processing in cloud federation requires a search and optimization techniques to find an optimal solution. This section describes the methods for both problems.

In MOO, general search and optimization techniques are classified into three categories: enumerative, deterministic, and stochastic [31] as follows:

- Enumerative,
- Deterministic,
 - Greedy,
 - Hill-Climbing,
 - Branch and Bound,
 - Depth-First,
 - Breadth-First,
 - Best-First (A^*, Z^*, \dots),
 - Calculus-Based,
- Stochastic,
 - Random Search,
 - Simulated Annealing,
 - Monte Carlo,

- Tabu Search,
- Evolutionary Computation.

The simplest search strategy is enumerative. The possible solution is evaluated in some predefined search space. However, it is inefficient when the space is large. In particular, a limiting search space should be implemented to find solutions in limited time [94].

The second class is a deterministic strategy. Deterministic algorithms try to integrate the knowledge of the problem in searching for solutions. There are several approaches in this class. Firstly, Greedy algorithms optimize solutions locally [21]. So, these algorithms fail in searching for global optimization. Hill-climbing algorithms search solutions in the direction of the best cost from the current position. However, the algorithm effectiveness is decreased in the problem of having local optima, plateaus, or ridges in the fitness (search) landscape [116]. Greedy and hill-climbing strategies are good in many cases. From a point, algorithms examine all directions to find the best step. Branch and bound strategies need decision algorithms to limit the search space [56]. The bounds are computed at a given point. After that, several points are compared and the most promising branch is determined [99]. Depth-first algorithms do not need searching for information. It generates all the possible successors, expands a successor, and so on. If the search is terminated, it restarts from the deepest node left behind [111]. If the node is unpromising, the backtracking is used to return the node's parent [99]. Breadth-first search is similar to depth-first. The difference between them is the actions after the node is generated. At the expansion node, breath-first explores more one layer [111]. Heuristic information is used to place values on an expanding node in best-first search. The best node is examined first [111]. The popular best-first search versions are A^* , Z^* , and others. They select a node to expand using the overall cost to get that node. Finally, calculus-based search algorithms use the minimum calculation to get optimal value [31].

Above algorithms are enumerative and deterministic methods. They are widely used in various problems [21, 99, 56, 111]. However, many MOPs are high dimensional or NP-Complete [31]. Deterministic algorithms are not suitable in NP-Complete or high dimensional problems [94, 56, 62].

The third class search algorithm includes Simulated Annealing (SA) [78], Monte Carlo methods [106], Tabu search [61], and Evolutionary Computation (EC) [62]. They are developed as alternative approaches for solving MOPs. Stochastic strategies need

a function to fitness values of solutions and a mapping mechanism between the problem and algorithm domain. This strategy does not guarantee an optimal solution. They provide a wider range of optimization problems, comparing to traditional deterministic search algorithms [62].

The simplest stochastic search strategy is Random search [31]. A given number of randomly selected solutions is evaluated simply. From a starting point, the next solution is evaluated randomly, called as a random walk [151]. Similar to enumeration methods, the algorithms are not suitable for MOPs. They do not integrate the knowledge into the problems. Random searches may generate no better result than enumerative algorithms [62].

An annealing analogy is used in the SA algorithm. While the hill-climbing selects the best move from a node, SA chooses a random step. If the current optimum is not improved by this moving, it would made some probability $p < 1$, else it is executed. This probability quickly decreases either by time or with the number of steps which the current optimum is declined [116].

Monte Carlo strategies are also stochastic methods. The random search of each step is independent of previous search [106, 117]. The comparator stores the current good solution and all the decision values. Tabu search is a strategy which helps to fall to local optima. It stores a record of solutions and the path reaching them. This strategy avoids the choice of solution to stuck on local optima. Tabu search is often implemented with other optimization algorithms [61, 117].

EC algorithms are stochastic search algorithms which simulate the natural evolutionary process. The key technique of EC are genetic algorithms (GAs), evolution strategies (ESs) and evolutionary programming (EP), as known as (EAs) [49]. These algorithms simulate the Darwinian concept of "Survival of Fittest" [62]. The common characteristic is a reproduction, random variation, competition and selection of individuals in the population [49]. An Evolutionary Algorithm (EA) has a population, a set of operators and some fitness functions to evaluate solutions. The fitness functions are used to determine the solutions are survived in the next generation.

The methods of deterministic or stochastic can be grouped into a group of mathematical programming. In this aspect, Linear programming is designed to solve problems in which all the relations are linear [68]. Non-linear programming methods are used to solve MOPs with convex constraint functions [117]. Stochastic programming methods use random parameters to solve MOPs. There are various methods depending on the

type of variables in the problems, such as discrete, integer, binary, mixed-integer programming [117]. As the classification of search and optimization techniques above, the following section presents the state-of-the-art of optimization problem in cloud environment.

3.3.1 Single Objective Optimization

In cloud computing, decision-making problems require several objectives: minimize monetary fees, maximize profit, minimize response time, space size of data, moving data between clouds, etc. The aim of Single-Objective (SO) optimization is to find the best solution which achieves the minimum or maximum value of a single objective function.

Many researches focus on Single-Objective Problem only. In particular, the execution time is the most considered metric in many systems. In this optimization problem, users have only one metric value to compare a candidate to another. However, in cloud federation, there are many metrics that users concern, e.g, the execution, the monetary, size of data storage, etc.

Some foundation results [26] show an overview of query optimization in relational systems. The cost estimation depends on statistical summaries of data and CPU, I/O, communication costs. The paper also shows the design of effective and correct SQL transformations is hard and building an extensible enumeration architecture is a significant undertaking. Besides, the limitations are that the optimization problems are SO problems and single system.

PIXIDA [80] provides a schedule to minimize data movement across multiple data centers. They integrate their algorithm in Spark and show that PIXIDA achieves a significant traffic reduction. The optimization problem they are concerned is a single objective optimization.

The heterogeneous database system [42] supports various database engines. The system could be built to store and optimize queries for the heterogeneous data systems. However, they use the single-objective model to optimize queries, not using MOQP algorithms for optimization and search problems.

OtterTune [147] is an automatic tuning method for DBMS configuration. The machine learning algorithms in Google TensorFlow [162] and Python scikit-learn [112] are used in the system. The target of the optimization problem is the total execution time

for OLAP and the latency for OLTP workload.

Non-Invasive Progressive Optimization for In-Memory Databases [157] introduces an approach to improve runtime, up to a factor of 4.5 compared to the worst case runtime. The method is able to exploit more properties than just the cardinality to re-optimize query plans and do not require any statistics over the data. However, they should integrate other relational operators into their optimization approach.

Lookahead Information Passing (LIP), a query execution strategy, is introduced to collapse the space of left-deep query plans for star schema warehouse down to a single point near the optimal plan [159]. To avoid a poor join order from the optimizer, they pass succinct filter data structures from the outer relations in all the joins to the inner relation. They implement the LIP technique in the RDBMS.

The join caching optimization is present in [24]. They show CJOIN, a new design for large-scale data analytics systems processing many concurrent join queries. They implement CJOIN as an extension to the PostgreSQL DBMS. CJOIN shares the common parts of the queries execution plans across multiple queries. However, they focus on a single system and mono objective optimization. It is not suitable for a heterogeneous database system and cloud federation.

Sharing the cost among users in the optimization process [146] is provided as Mechanism Design. Multiple users accessing the service for the same time period is the target of the system. They built a series of mechanisms to share pricing in either an off-line or an on-line game. The solution is especially good in the case that many users derive significant value from an optimization during the same time-slot. However, they do not consider MOOPs.

Cost modeling can be built without machine learning algorithms. For example, a cost model in [153] is introduced to build an optimal group of queries in PostgreSQL. This cost model aims to predict query execution time for this specific relational Data Base Management system. However, in a cloud federation with variability and different systems, cost functions may be quite complex.

The investigation of query performance [88] shows that the impact of the cost model is less influential than the cardinality estimation. They focus on the technique to improve performances despite the sub-optimal cardinality estimation. However, their method focus on a specific DBMS, such as PostgreSQL, and the cost models are limited in these database engines.

However, SO optimization does not solve the problem of multiple objectives. Most

of the recent research often solve by turning MOP into a single-objective problem first and then solving it. The following section shows the method researches use to optimize a MOP.

Weighted Sum Model

Weighted Sum Model is useful as a tool which should provide decision makers with insights into the nature of the problem. Some recent researches use Weighted Sum Model in query processing and optimization process.

The R-skyline [28] operators generalize and unify skyline and ranking queries. They focus on choosing weights for a scoring function. Their experiments show that non-dominated and potentially optimal are very effective in focusing on tuples of interest. However, the application is only for MOOP in a single database system.

A homogeneous cloud environment is considered in a MOP [92]. The system is designed in a scenario of multisite clouds, a cloud with multiple data centers in different locations. Two cost models including execution time and monetary costs are used to estimate the cost of executing scientific workflows. The MOOP is solved by grouping two cost metric into a scalar value by the weights for execution time and monetary costs. However, the system does not consider the cloud federation and variability of cloud environment. Besides, MOOP is converted to a weighted sum model [67]. The estimation of weights corresponding to different objectives in this model is also a MOOP. In addition, the objectives may be contradictory. For example, Killapi [79] stated that cloud providers lease computing resources that are typically charged based on a per time quantum pricing scheme. The solutions represent trade-offs between time and money. Hence the execution time and the monetary cost cannot be homogeneous.

Furthermore, 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 [50]. The research in [75] 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. As a consequence, MOOPs lead to find solutions by Pareto dominance techniques.

3.3.2 Multiple Objective Optimization

As the comment in [60], SO problems cannot adequately represent Multi-Objective (MO). This approach may significantly change the problem of nature. In some cases, the problem becomes harder to solve, or certain optimal solutions are not found anymore [60]. In general, MO Optimization problem is more complex than SO Optimization problem. Moreover, the large space of candidates leads to the necessity of finding a Pareto set of a solution in MOOP. Besides, generating Pareto-optimal front is often infeasible due to high complexity [160]. The next section shows various methods in finding a near optimal or approximations.

Near Optimal

The first paper [145] experiments the multiple query optimization problems on a quantum computer published in the database community. The quantum annealer is produced by the Canadian company D-Wave. They compare many MOO algorithms, such as integer linear programming [43], genetic algorithm [12] and iterated hill climbing [44] in the experiments. The papers show that the genetic algorithm is better than the hill climbing algorithm and the hill climbing algorithm often produces slightly better solutions than the linear solver. Besides, experiments show that the quantum annealer finds near-optimal faster than various classical optimization approaches.

The group-based Skyline (G-Skyline) [156] is a good option for finding a group solution for the multiple objective problem. Their method is built to find G-Skyline groups using DSG (Directed Skyline Graph). However, a user is often concerned the optimal point in a Pareto set of solutions. Hence, the group with k-level in this method has many solutions he does not consider.

Meanwhile, Evolutionary Algorithms, an alternative to the Pareto-optimal, look for approximations (set of solutions close to the optimal front). For example, Evolutionary Multi-objective Optimization (EMO) approaches [72, 81, 161, 40, 39, 122] have been developed based on Pareto dominance techniques.

Among EMO approaches, [40, 37] proposed Non-dominated Sorting Genetic Algorithms (NSGAs) to decrease the computational complexity while maintaining the diversity among solutions. The crowding distance operators are used to maintain the diversity in NSGA-II [40] and SPEA-II [161]. However, the crowding distance operators need to be replaced because of high complexity and not suitability for the problems of

more than two objectives [76]. Furthermore, MOEA/D maintains the diversity with more than three objectives problem [158]. This algorithm uses an approach based on decomposition, a basic strategy in traditional multi-objective optimization, to divide a multiple objectives problem into various single objective optimization sub-problems. Nevertheless, MOEA/D can only solve up to four objectives [118]. Meanwhile, Deb and Jain [39] proposed a set of reference directions to guide the search process in NSGA-III. In spite of good quality, NSGA-III has the highest computational complexity among NSGAs.

NSGA-II is often applied to MOP in cloud environment. In particular, the framework Flower [77] collects information from multiple monitoring systems of the cluster at different layers. After that, the control system takes them as input the history of the sensor values. The desired values are estimated at a specific time. Flower finds an optimized solution by a multi-objective genetic algorithm, NSGA-II [40], to efficiently search the provisioning plan space.

As a consequence, EMOs are appropriated for finding alternative Pareto-optimal and NSGAs show advantage among Evolutionary Multi-Objective Algorithms class. Although NSGAs are used in many research works to find an approximate optimal solution for MOOP, NSGAs still need to be improve their quality.

3.4 Conclusion

Medical data in cloud federations are required to face challenges in data management in terms of Multi-Objective Optimization in hybrid data storage configuration, query processing, heterogeneous database engines.

Therefore, the main goals of our study are to propose a solution for the ever-increasing size, high velocity and variety of medical data. In particular, our solution provides an efficient approach to find a DICOM optimal data layout in terms of both data storage and query processing strategy for OLTP and OLAP queries. Besides, we also propose a model of cloud federation to organize the data in heterogeneous database engines in a cloud environment.

MULTI-OBJECTIVE OPTIMIZATION

Contents

4.1 Introduction	47
4.2 Pareto set	48
4.3 Multiple Linear Regression	49
4.3.1 Linear Regression	50
4.3.2 Multiple Linear Regression	53
4.4 Non-dominated Sorting Genetic Algorithm	57
4.4.1 NSGA process	57
4.4.2 Application	60
4.5 Conclusion	61

4.1 Introduction

As we recommended in Chapter 2, most of solutions for clouds do not consider MOOP. Only the IReS open source platform considers heterogeneous data bases, the variability of the cloud environment, but all the information in historic data is used for the estimation cost values during the optimization process. They may have expired at the moment of optimization processing. Besides, medical data management in cloud environment, as described in Chapter 3, needs an efficient MOO algorithm to optimize data storage configuration in clouds.

Among Multi-Objective Optimization algorithm classes, EMO shows their advantages in searching and optimizing for the MOP. Non-dominated Sorting Genetic Algorithms (NSGAs) are often used to decrease the computational complexity while maintaining the diversity among solutions [39, 142], but their quality should be improved. This chapter will present the background of the estimation using machine learning approach and NSGAs. We first show the definition of Pareto set in Section 4.2. Next, in

Section 4.3, we introduce the background of our estimation approach using machine learning algorithms. In Section 4.4, we introduce the process of NSGAs. Finally, we conclude with Section 4.5.

4.2 Pareto set

Pareto dominance techniques are often used in MOO algorithms, such as EMO [72, 81, 161, 40, 39, 122, 158]. In the vast space of solution candidates of MOO approach, a candidate 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 MOP.

In particular, in a query processing problem, let a **query** q be an information request from databases, presented by a set \mathcal{Q} of tables. A **Query Execution Plan** (QEP) includes an ordered set of operators (select, project, join, etc.). The set of QEPs p of q is denoted by symbol \mathcal{P} . The set of operators is denoted by \mathcal{O} . A plan 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. Their 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} . In Multi-Objective Query Processing (MOQP), N denotes 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 \mathbf{C} be denoted 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:

$$\mathbf{C}(p_1, \mathcal{X}) \preceq \mathbf{C}(p_2, \mathcal{X}) \mid \forall n \in N, \forall \mathbf{x} \in \mathcal{X} : c^n(p_1, \mathbf{x}) \leq c^n(p_2, \mathbf{x}). \quad (4.1)$$

The function $Dom(p_1, p_2) \subseteq \mathcal{X}$ yields the parameter space region where p_1 dominates p_2 [144]:

$$Dom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) \leq c^n(p_2, x)\}. \quad (4.2)$$

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 [144], i.e.

$$StriDom(p_1, p_2) = \{x \in \mathcal{X} \mid \forall n \in N : c^n(p_1, x) < c^n(p_2, x)\}. \quad (4.3)$$

A **Pareto region** of a plan is a sub-space of plans. There is no alternative plan in the rest of the plans which is not better than the plans in the **Pareto region** [144]:

$$PaReg(p) = \mathcal{X} \setminus \left(\bigcup_{p^* \in \mathcal{P}} StriDom(p^*, p) \right). \quad (4.4)$$

For example, three query plans (QEPs) with their costs for monetary costs, execution time of query Q are shown in Table 4.1. All three QEPs are in Pareto plan set of Q . QEP3 is not the best plan neither Time or Monetary cost, but QEP1 and QEP2 are not dominated QEP3 in both of objectives. Hence, QEP3 belongs to the Pareto plan set of Q .

Table 4.1 – Multiple Objectives for Query Execution Plans

QEP	Vms	Price (\$/60min)	Time (min)	Monetary (\$)
QEP1	10	0.02	60	0.2
QEP2	40	0.02	22	0.29
QEP3	30	0.02	26	0.26

4.3 Multiple Linear Regression

In many database management systems, predicting cost values is useful in optimization process [153]. Recent research has 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 [121]. In this chapter, 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, \dots , and x_L . For example, execution time c is a function of data size x_1 of first element in a 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 $x_i, i = 1, 2, \dots, L$. We focus in the estimation the relationship between c and the independent variables x_1, x_2, \dots , and x_L based on this sample. The following sections introduce the simple defined cases above.

4.3.1 Linear Regression

Assuming that a variable c is a function of an independent variable and the relationship between them is linear. The relationship means that the mean of c is, $E\{c\}$, is known to be linear function of x , that is,

$$E\{c\} = \beta_0 + \beta_1 x, \quad (4.5)$$

The intercept β_0 and slope β_1 constants are unknown. They are estimated from a sample of c values with their associated values of x . Random variable c is a function of x . Defining a random variable E by

$$E = c - (\beta_0 + \beta_1 x), \quad (4.6)$$

the equation becomes

$$c = \beta_0 + \beta_1 x - E, \quad (4.7)$$

where E has mean 0 and variance σ^2 , which is identical to the variance of c . The value of σ^2 is not known and not a function of x .

The simple linear regression model is shown in Equation 4.7. The unknown parameters β_0, β_1 are regression coefficients. The random variable E presents the mean deviation of c .

Least square method of estimation

The target of this estimation approach is how the sum of the square of the differences between observed sample values c_i and the estimated expected value of c , $\hat{\beta}_0 + \hat{\beta}_1 x$, where $\hat{\beta}_0, \hat{\beta}_1$ are the estimation of the regression parameter β_0, β_1 , respectively. The difference between a sample and an estimated expected value is calculated as follows:

$$e_i = c_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i). \quad (4.8)$$

The least-square estimates $\hat{\beta}_0, \hat{\beta}_1$ are defined by minimizing

$$Q = \sum_{i=1}^m e_i^2 = \sum_{i=1}^m (c_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2. \quad (4.9)$$

Assuming that we have the observed sample values $(x_1, c_1), (x_2, c_2), \dots, (x_m, c_m)$, the observed regression equation is

$$c_i = \beta_0 + \beta_1 x_i + e_i; i = 1, \dots, m. \quad (4.10)$$

Let denote

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_m \end{bmatrix}, \quad (4.11)$$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_m \end{bmatrix}, \quad (4.12)$$

$$E = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ e_m \end{bmatrix}, \quad (4.13)$$

$$B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}. \quad (4.14)$$

The matrix version of Equation 4.10 is

$$C = AB + E. \quad (4.15)$$

The sum of squared residuals of Equation 4.9 is rewritten

$$Q = E^T E = (C - AB)^T (C - AB). \quad (4.16)$$

Setting $\delta Q = 0$, the solution for \hat{B} is obtained from normal equation

$$\delta Q = -\delta B^T A^T (C - AB) - (C - AB)^T A \delta B = 0$$

$$\delta Q = -2\delta B^T A^T (C - AB) = 0,$$

or

$$A^T AB = A^T C. \quad (4.17)$$

Hence, the solution for \hat{B} is

$$\hat{B} = (A^T A)^{-1} A^T C. \quad (4.18)$$

Coefficient determination

Assuming that the fitted value $\hat{c}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, i = 1, \dots, m$, the coefficient of determination [74, 113] is defined by

$$R^2 = 1 - \frac{SSE}{SST}, \quad (4.19)$$

where SSE is the sum of squared errors and SST is the total sum of squares following

$$SSE = \sum_{i=1}^m (c_i - \hat{c}_i)^2, \quad (4.20)$$

$$SST = \sum_{i=1}^m (c_i - \mu_c)^2, \quad (4.21)$$

and the mean of observed value is

$$\mu_c = \frac{\sum_{i=1}^m c_i}{m}. \quad (4.22)$$

The coefficient of determination is always between 0 and 1. R^2 represents the proportion of total variation in the response variable. For example, the model $c = \hat{\beta}_0 + \hat{\beta}_1 x$ give $R^2 = 0.75$, it can be concluded that 3/4 of variation in fitted value can be explained by the linear relationship between the input variables and fitted values.

4.3.2 Multiple Linear Regression

The Linear Regression above is used to estimate a cost value. In MOOP, we need to extend this approach to estimate various cost functions. Multiple Linear Regression is used in this situation to estimate multiple cost values.

A cost function of Multiple Linear Regression (MLR) model [121] is defined as follows:

$$c = \beta_0 + \beta_1 x_1 + \dots + \beta_L x_L + e, \quad (4.23)$$

where $\beta_l, l = 0, \dots, L$, are unknown coefficients, $x_l, l = 1, \dots, L$, are the variables, e.g., size of data, computer configuration, etc., c is cost function values and e is random error following normal distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2 . The **fitted equation** is defined by:

$$\hat{c} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_L x_L. \quad (4.24)$$

Example 4.3.1 A query **Q** 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 database engine [129], while GeneralInfo table is in cloud B with PostgreSQL database engine [133]. This scenario leads to consider 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 previous data, each of them associates with a response c_m , the system

has observed regression equations:

$$C = \beta_0 + \beta_1 x_1 + \dots + \beta_L x_L + E, \quad (4.25)$$

where x_l is a set value of variable $x_{lm}, l = 1, \dots, L, m = 1, \dots, M, C$ is a set value of observed value $c_m, m = 1, \dots, M,$ and E is the random error, with mean 0 and variance σ^2 .

Least squares method of estimation

The target of this method is estimating the regression coefficients. Given observed sample value sets $(x_{1i}, x_{2i}, \dots, x_{mi}, c_i), i = 1, \dots, M,$ the system of observed regression equations is

$$c_m = \beta_0 + \beta_1 x_{1m} + \dots + \beta_L x_{Lm} + e_m; m = 1, \dots, M. \quad (4.26)$$

Let denote

$$A = \begin{bmatrix} 1 & x_{11} & x_{21} & \dots & x_{L1} \\ 1 & x_{12} & x_{22} & \dots & x_{L2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_{1M} & x_{2M} & \dots & x_{LM} \end{bmatrix}, \quad (4.27)$$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_M \end{bmatrix}, \quad (4.28)$$

$$E = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ e_M \end{bmatrix}, \quad (4.29)$$

$$B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \cdot \\ \cdot \\ \beta_L \end{bmatrix}. \quad (4.30)$$

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

$$SSE = \sum_{m=1}^M (c_m - \hat{c}_m)^2, \quad (4.31)$$

The matrix version of Equation 4.26 is

$$C = AB + E. \quad (4.32)$$

The sum of squared residuals of Equation 4.31 is rewritten

$$Q = E^T E = (C - AB)^T (C - AB). \quad (4.33)$$

Setting $\delta Q = 0$, the solution for \hat{B} is obtained from normal equation

$$\delta Q = -\delta B^T A^T (C - AB) - (C - AB)^T A \delta B = 0$$

$$\delta Q = -2\delta B^T A^T (C - AB) = 0,$$

or

$$A^T AB = A^T C. \quad (4.34)$$

Hence, the solution for \hat{B} is retrieved by

$$\hat{B} = (A^T A)^{-1} A^T C. \quad (4.35)$$

Other regression models

The multiple linear regression approach can be extended to build regression models which are nonlinear in the independent variables. Following equations show examples

of nonlinear regression models

$$c = \beta_0 + \beta_1 x^2 + \beta_2 x^3 + e, \quad (4.36)$$

$$c = \beta_0 + \beta_1 x_1^2 + \beta_2 x_2^2 + \beta_3 x_1 x_2 + e, \quad (4.37)$$

$$c = \beta_0 + \beta_1 \sin(x) + e, \quad (4.38)$$

$$c = \beta_0 + \beta_1 \exp(x) + e, \quad (4.39)$$

$$c = \beta_0 + \beta_1 \log(x) + e, \quad (4.40)$$

Equation 4.36 and Equation 4.37 are polynomial models. They are still the linear in the unknown parameters, $\beta_0, \beta_1, \beta_2, \dots, etc..$ The multiple linear regression can be applied to the models. In particular, let $x_1 = x^2$, and $x_2 = x^3$ in Equation 4.36, the model becomes a multiple linear regression model with two independent variables. Equation 4.37 can be a multiple linear regression with four independent variables by the same method in Equation 4.36. Other models can be converted to a linear regression model, such as, 4.39, 4.38, 4.40 with letting $x_1 = \sin(x)$, or $x_1 = \exp(x)$, or $x_1 = \log(x)$.

Multiple Linear Regression is the basic model we use to develop our estimation method for estimating cost value of each candidate of a MOOP in the cloud federations. Besides of estimation process, MOOPs also need to use a MOO algorithm to select an appropriate candidate. It leads to finding solutions by Pareto dominance techniques. Because of high complexity [160], MOOP leads to finding an approximate optimal solution by Pareto dominance techniques. A well known approach to solve the high complexity of MOOP is EMO. Among EMO approaches, Non-dominated Sorting Genetic Algorithms (NSGAs) [40, 37] have lower computational complexity than other EMO approaches [40]. The next section describes the basic principle of NSGAs.

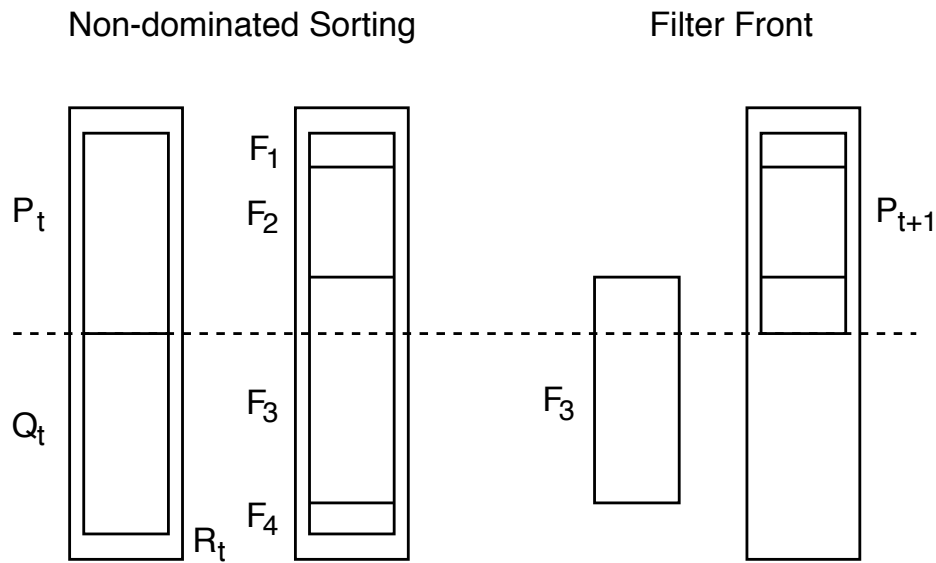


Figure 4.1 – NSGA-II and NSGA-III procedure [40, 39].

4.4 Non-dominated Sorting Genetic Algorithm

Among MOO algorithm classes, as described in Chapter 3, EMO shows their advantages in searching and optimizing for the MOP. Among EMO approaches, Non-dominated Sorting Genetic Algorithms provide low computational complexity of non-dominated sorting, $O(MN^2)$ of NSGAs comparing to $O(MN^3)$ of other EMO, where M is the number of objectives and N is the population size.

4.4.1 NSGA process

Initially, NSGAs start with a population P_0 consisting of N solutions. In the hybrid data optimization problem, a population represents a set of candidates of hybrid data storage configuration. The size of P_0 is smaller than the space of all candidates. 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 [38], 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 $R_0 = P_0 \cup Q_0$

Algorithm 1 Generation t of NSGA-II and NSGA-III [40, 39].

```

1: function EVALUATION( $P_t, N$ )
2:    $S_t = 0, i = 1$ 
3:    $Q_t = \text{Recombination} + \text{Mutation}(P_t)$ 
4:    $R_t = P_t \cup Q_t$ 
5:    $\mathcal{F}_1, \mathcal{F}_2, \dots = \text{Non-diminated-sort}(R_t)$ 
6:   while  $|S_t| \leq N$  do
7:      $S_t = S_t \cup \mathcal{F}_i$ 
8:      $i++$ 
9:   end while
10:  Last front is  $\mathcal{F}_l$ 
11:  if  $|S_t| = N$  then
12:     $P_{t+1} = S_t$ 
13:    break
14:  else
15:    select  $N - \sum_{j=1}^{l-1} |\mathcal{F}_j|$  solutions in  $\mathcal{F}_l$ 
16:  end if
17:  return  $P_{t+1}$ 
18: end function

```

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 the usual domination principle [25] 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.

NSGA-II [40] and NSGA-III [39] follow the same process, illustrated by Algorithm 1. The procedure is illustrated in Figure 4.1. Assume that the process is at the t^{th} generation. A combined population $R_t = P_t \cup Q_t$ is formed. The population R_t is sorted in level $\mathcal{F}_1, \mathcal{F}_2, \text{etc.}$ Solutions in the best non-dominated \mathcal{F}_1 are good solutions in R_t . If the size of \mathcal{F}_1 is smaller than N , all members of \mathcal{F}_1 are selected to P_{t+1} . Thus, solutions in \mathcal{F}_2 are chosen next and so on. This process continues until no more level can be fitted in p_{t+1} . The last level \mathcal{F}_l cannot fill in P_{t+1} , *i.e.* $\sum_{j=1}^l |\mathcal{F}_j| > N$.

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 [40] and SPEA-II [161] 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 [84, 71]. Hence, when the population has a high-density area, higher than

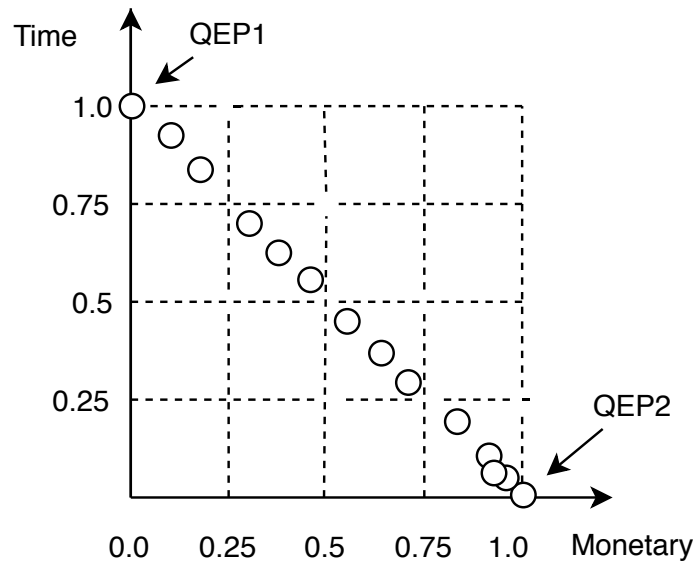


Figure 4.2 – An example of using the crowding distance in NSGA-II.

others, NSGA-II prefers the solution which is located in a less crowded region. For example, when the size of the population is 10, NSGA-II rejects four solutions which are near to point $(1.0, 0.0)$, as illustrated in Figure 4.2.

On the other hand, MOEA/D [158] 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 the good neighborhood is not mentioned. The diversity is considered as the selected solution associated with these different sub-problems. Experimental results in [39] 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 [39] (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 [39], 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 Figure 4.2. After selection process, the diversity of population is better than NSGA-II with solutions close to three reference points. How-

ever, 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}_i . Therefore, when the number of solutions or objectives is significant, the time for calculating and comparing is considerable.

4.4.2 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 providers usually leases computing resources that are typically charged based on a per time quantum pricing scheme [79]. 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 single-objective problem. Moreover, if we want to reduce the MOOP to a SOOP, we should have a policy to group all objectives by the Weighted Sum Model (WSM) [67]. However, estimating the weights corresponding to different objectives in this model is also a multi-objective problem.

In addition, MOOPs could be solved by MOO algorithms or WSM [67]. 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 [50]. Furthermore, the research in [75] 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 find a Pareto-optimal solution.

In conclusion, MOOP leads to using Pareto dominance techniques. A pareto-optimal front is often infeasible [160]. NSGAs show the advantage in searching a Pareto solution for MOOP in less computational complexity than other EMO [40]. However, they should be improved the quality of solving MOP when the number of objectives is significant.

4.5 Conclusion

This chapter concludes the MOOP background we use to solve the MOP in a cloud environment. First, Pareto set aspect is also presented for the MOOP. After that Multiple Linear Regression is introduced to build a cost model for estimating multiple cost values in MOP. This technique is widely used in science and engineering [72, 81, 161, 40, 39, 122, 158]. Finally, we focus on the process of a good technique in EMO class. In particular, NSGAs show their advantage in EMO algorithms. However, the characteristics of NSGAs should be improved to have better performance, such as diversity, convergence.

Recent research shows all their advantages and disadvantage in various areas in cloud environment. Chapter 2 presents solutions for clouds, however only IReS open source platform consider the heterogeneous problem and can be extended to optimize solve MOP. Chapter 3 presents various approaches to optimize data storage configuration of medical data. However, only [97] consider the characteristic of sparse and workload of DICOM data. Nevertheless, the authors do not provide an optimal solution for hybrid data storage configuration of DICOM. Chapter 4 describes the recent research in estimation cost value and optimization approaches for MOP. The historical information should be used efficiently in machine learning approaches. Besides, NSGAs should be improved quality for MOP, when the number of objectives is significant.

The next chapters will show our approaches to solve the problem of medical data in cloud federations. First, the heterogeneity should be solved when the system connects data from various database engines in the clouds. Second, the variability of cloud environment requires the estimation process to be more efficient. Third, searching and optimizing a solution in MOOP should be solved by an efficient MOO algorithm. This will make possible not only to process queries but also to find an optimal solution of hybrid data storage configuration.

Key Points

- We introduced the background and the existing solutions in clouds and cloud federations.
- We presented an overview of medical data management on clouds. Search and optimization techniques are also described as the state-of-the-art of optimization problem in cloud environment.
- We showed the background of the estimation using machine learning approach and Non-dominated Sorting Genetic Algorithms solutions.

PART II

Techniques for cloud federation

DYNAMIC REGRESSION ALGORITHM

Contents

5.1 Introduction	64
5.2 Problem	64
5.3 DREAM	67
5.3.1 Coefficient of determination	67
5.3.2 Cost Value Estmation	69
5.3.3 Optimization	70
5.4 Conclusion	72

5.1 Introduction

The aspects and background related to our problem in the cloud federation are introduced in previous chapters, including Multiple Linear Regression, Non-dominated Sorting Genetic Algorithm, Hybrid data configuration, heterogeneous database engines, Multi-Objective Problems and cloud environment.

This chapter presents the first contribution, which is the estimation of accurate cost values in the variable environment of a cloud federation. We first show in Section 5.2 the problem of estimation. Next, our algorithm is proposed in Section 5.3. Finally, we conclude with Section 5.4.

5.2 Problem

Most cost models [103, 153, 48] depend on the size of data. In particular, cost function and **fitted value** of Multiple Linear Regression model are previously defined in

Equation 4.24. To remind the definition, a cost function of Multiple Linear Regression model [121] is defined as follows:

$$c = \beta_0 + \beta_1 x_1 + \dots + \beta_L x_L + e, \quad (5.1)$$

where $\beta_l, l = 0, \dots, L$, are unknown coefficients, $x_l, l = 1, \dots, L$, are the variables, e.g., size of data, computer configuration, etc., c is cost function values and e is random error following normal distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2 . The **fitted value** is defined by:

$$\hat{c} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_L x_L. \quad (5.2)$$

There are M data samples, each of them associated with a response c_m . The system has observed regression equations:

$$C = \beta_0 + \beta_1 x_1 + \dots + \beta_L x_L + E, \quad (5.3)$$

where x_l is a set value of variable $x_{lm}, l = 1, \dots, L, m = 1, \dots, M$, C is a set value of observed value $c_m, m = 1, \dots, M$, and E is the random error, with mean 0 and variance σ^2 .

Let us denote

$$A = \begin{bmatrix} 1 & x_{11} & x_{21} & \dots & x_{L1} \\ 1 & x_{12} & x_{22} & \dots & x_{L2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_{1M} & x_{2M} & \dots & x_{LM} \end{bmatrix}, \quad (5.4)$$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_M \end{bmatrix}, \quad (5.5)$$

$$E = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ e_M \end{bmatrix}, \quad (5.6)$$

$$B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \cdot \\ \cdot \\ \beta_L \end{bmatrix}. \quad (5.7)$$

The solution for \hat{B} in [121] is retrieved by

$$\hat{B} = (A^T A)^{-1} A^T C. \quad (5.8)$$

The bigger M for sets $\{c_m, x_{lm}\}$ is, the more accurate MLR model usually is. However, the computers is slowing down when M is too big.

Example 5.2.1 *Assuming that a query is processed on Amazon EC2. If the pool of resources includes 70 vCPU, 260GB of memory, the minimum of memory for a configuration is 1GB and the memory size is a multiple of 1GB, the number of different configurations to execute this query is thus $70 \times 260 = 18,200$. Hence, the system can generate 18,200 equivalent QEPs from a given execution plan.*

Example 5.2.1 shows that a query execution plan can generate multiple equivalent QEPs in cloud environment. The smaller M for sets $\{c_m, x_{lm}\}$ is, the faster the estimation cost process of Multi-Objective Query Processing for a QEP is. In Example 5.2.1, even a small reduction in the computation cost for the estimation of a single QEP can result in a high reduction of the cost for the estimation of hundreds or thousands of similar/equivalent QEP.

Furthermore, the target of Multi-Objective Query Processing is MOOP [158], which is defined by:

$$\text{minimize}(F(x) = (f_1(x), f_2(x), \dots, f_K(x))^T), \quad (5.9)$$

where $x = (x_1, \dots, x_L)^T \in \Omega \subseteq \mathbb{R}^L$ 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, there is no point in Ω that minimizes all the objectives together. As mentioned in Chapter 4, 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 [11]. In cloud environment, the number of equivalent query execution plans is multiplied.

5.3 DREAM

Machine learning algorithms often use entire historic datasets in training and testing cost models. However, the cloud federation is such a variable environment that even small changes in the data, cluster configurations, or network will make a difference in cost models. The entire historic datasets does not reflect the updated characteristic of the systems. The expired data maybe affect to the accuracy of cost models using the entire historic datasets. For example, the old cluster with 5 instances (each instance has 1 vCPU and 2GB of memory) is replaced by a new one with 10 instances (each instance has 4 vCPU and 8 GB of memory). The information of the cluster should be updated and the old data is expired.

5.3.1 Coefficient of determination

The most important idea in our algorithm is to estimate MLR quality by using the coefficient of determination. The coefficient of determination [121] is defined by:

$$R^2 = 1 - SSE/SST, \quad (5.10)$$

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 variance accounted for using the Multiple Linear Regression model and variable X . For example, if the model gives $R^2 = 0.75$ of response time cost, we can conclude that 3/4 of the variation in response time values can be explained by the linear relationship between the input variables and response time cost. Table 5.1 presents MLR with different number of measurements. The smallest dataset is $M = L + 2 = 4$ [121], where M is the size of previous data and L is the number of variables in (4.23). In general, R^2 increases proportionally with M . However, if the model requires $R_{require}^2$ should be greater than 0.8 to provide a sufficient quality of service level. As a consequence, M should be greater than or equal to 6 to provide enough accuracy.

Table 5.1 – Using MLR in different size of dataset.

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

In some cases, the system requires the minimum values of R^2 is equal to 0.8, $M > 7$ is not recommended because of the large size of historical datasets. 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.

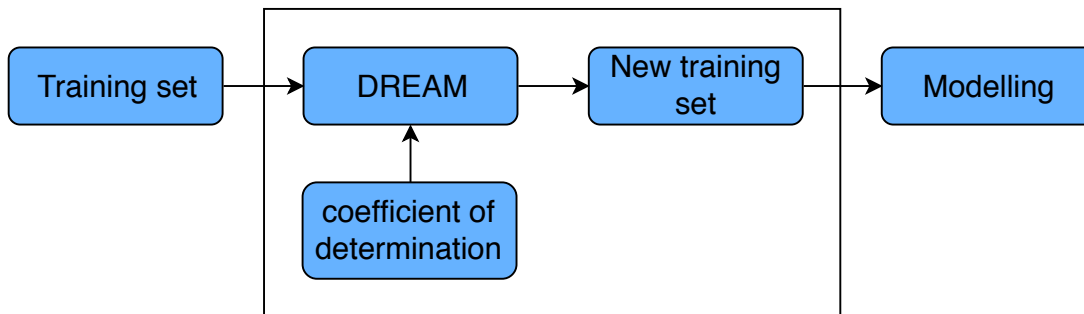


Figure 5.1 – DREAM module.

Our motivation is to provide accurate estimation while reducing the size of historical datasets based on R^2 . We thus propose DREAM as a solution for cloud federation and their inherent variance, as shown in Figure 5.1. 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 less data in building model process than the original approach.

5.3.2 Cost Value Estimation

Our algorithm uses the size of data as variables of DREAM. In (4.24), \hat{c} is the cost value, which needs to be estimated in MOQP, and $x_1, x_2, etc.$ are the characteristics of system, such as size of input data, the number of nodes, the type of virtual machines.

The sufficient quality of a model is determined by the datasets. However, the coefficient of determination is defined by trade-offs between the high value of R^2 and the small size of dataset. Hence, the coefficient of determination should be predefined by users. If $R^2 \geq R_{require}^2$, where $R_{require}^2$ is predefined by users, the model is reliable. In contrast, it is necessary to increase the number of set value. Algorithm 2 shows a scheme as an example of increasing value set: $m = m + 1$.

Algorithm 2 Calculate the predict value of multi-cost function

```

1: function ESTIMATECOSTVALUE( $R_{require}^2, X, M_{max}$ )
2:   for  $n = 1$  to  $N$  do
3:      $R_n^2 \leftarrow \emptyset$  //with all cost function
4:   end for
5:    $m = L + 2$  //at least  $m = L + 2$ 
6:   while (any  $R_n^2 < R_{n-require}^2$ ) and  $m < M_{max}$  do
7:     for  $\hat{c}_n(p) \subseteq \hat{c}_N(p)$  do
8:        $\hat{\beta}_n = (A_n^T A_n)^{-1} A_n^T C_n$  //estimate unknown coefficient
9:        $R_n^2 = 1 - SSE/SST$ 
10:       $\hat{c}_n = \hat{\beta}_{n0} + \hat{\beta}_{n1}x_1 + \dots + \hat{\beta}_{nL}x_L$ 
11:    end for
12:     $m = m + 1$ 
13:  end while
14:  return  $\hat{c}_N(p)$ 
15: end function

```

First, *EstimateCostValue* function requires the coefficient of determination $R_{require}^2$, historic data X , and the maximum value of observation window M_{max} . Normally, M_{max} is equal to the size of X . At initial step in line 2, the algorithm resets all the value of R^2 , and the size of data is starts at the value of $L+2$ as the smallest dataset [121], as shown in line 5. The *while* loop from line 6 to line 13 show that the matrix of coefficient $\hat{\beta}_n$, cost function \hat{c}_n , and the coefficient of determination R_n^2 are determined continuously until all R_n^2 are bigger than or equal to $R_{n-require}^2$, where $R_{n-require}^2$ is predefined by users for the cost function \hat{c}_n . The while loop is also stopped when the size of historic data m is equal to the maximum of observation window M_{max} . The scheme of increasing value

of m is shown in line 12. This is a simple example of changing the size of observation window. In particular, we can have other schemes of changing the size of observation window. For example, we can change the size of window faster than that by $m = 2 * m$. At the end, the cost function $\hat{c}_N(\mathbf{p})$ is used for estimate the value of objectives for MOP. In the scope of this thesis, we do not discuss about the best scheme of changing the size of observation window.

5.3.3 Optimization

In this section, 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 [67] with user policy to find the best candidate. However, Multi-objective Optimization algorithms have more advantages than WSM [50, 75]. As mentioned in Chapter 4, MOO algorithms are selected thanks to their advantages when comparing with WSM. Estimating the weights corresponding to different objectives in this model is also a multi-objective problem. Besides, the optimal solution of WSM could be unacceptable, because of an inappropriate setting of the coefficients [50]. Furthermore, the research in [75] 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. For example, as shown in Figure 5.2, two Multi-Objective Query Problem (MOQP) approaches are shown. If the weight sum model is changed, the new optimization process of Multi-Objective Optimization based on Genetic Algorithm can restart at the input of Weight Sum Model Values step. Otherwise, the system should restart at the beginning of optimization process of Multi-Objective Optimization based on Weighted Sum Model. Hence, our system applies a Multi-objective Optimization algorithm to find a Pareto-optimal solution.

Hence, after using DREAM, a Multi-objective Optimization algorithm, such as Non-dominated Sorting Genetic Algorithm II [40] is applied to determine a Pareto plan set. At the final step, the weight sum model \mathbf{S} and the constraint \mathbf{B} associated with the user policy are used to return the best QEP for the given query [67]. In particular, the most meaningful plan will be selected by comparing function values with weight parameters between \hat{c}_n [67] at the final step, as shown in Algorithm 3. Figure 5.2 shows the difference between these MOQP approaches. As shown in Figure 5.2, the Multi-

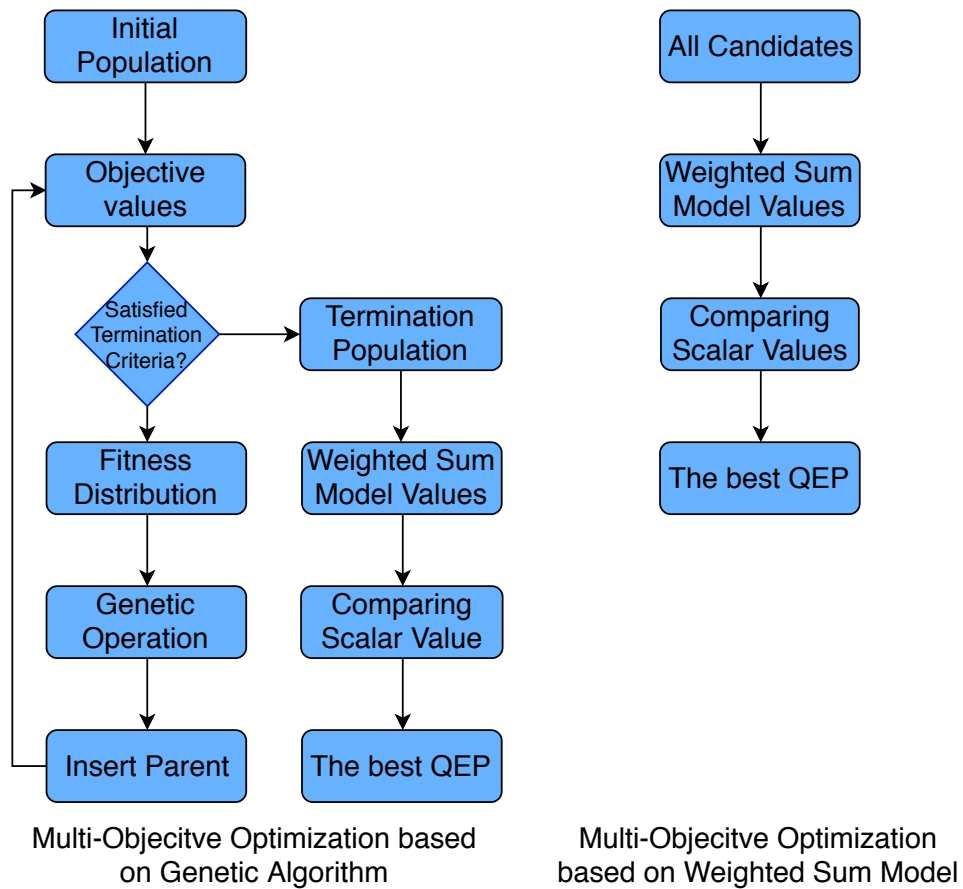


Figure 5.2 – Comparing two MOQP approaches

Objective Optimization based on Weighted Sum Model should evaluate all the space of candidates to find the best Query Execution Plan (QEP) of a given plan. The candidate space maybe very large to search the best QEP. Besides, when the WSM changes, the new optimization process is restarted at the beginning. Otherwise, the Multi-Objective Optimization based on Genetic Algorithms can restart at the input of Weight Sum Model Values step. The space of candidate at this step is equal to the size of population of Genetic Algorithm. Our algorithms are developed based on MLR described above using x_i for size of data and c_i for the metric cost, such as the execution time.

Function *BestInPareto* is used to choose the best candidate in the set of non-dominated candidates, called Pareto set. At the final step of MOOP, we have a Pareto set of candidates. Depending on a user demand, we have a weighted sum model **S** and constraints **B**. At the beginning, all candidates which have cost function values smaller than the value of constraints are added to the new set of non-dominated candidates,

Algorithm 3 Select the best query plan in \mathcal{P}

```

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  $p \in P_B | C(p) = \min(\text{WeightSum}(P_B, \mathbf{S}))$ 
5:   else
6:     return  $p \in \mathcal{P} | C(p) = \min(\text{WeightSum}(\mathcal{P}, \mathbf{S}))$ 
7:   end if
8: end function

```

as shown in line 2. If this set is not empty, line 3, the scalar value of each candidate is determined by the weighted sum model \mathbf{S} and the best solution is selected by the minimum value of $C(p)$, as shown in line 4. In contrast, when the size of new non-dominated candidates set is zero, the best candidate is chosen in the original Pareto set by weighted sum model \mathbf{S} , as shown in line 6. Finally, the best solution in the Pareto set \mathcal{P} is selected by the user with a weighted sum model S and a set of constraints B .

5.4 Conclusion

In this chapter, we have presented our contribution, DREAM, to improve the accuracy of the estimation of cost values in cloud federation. The size of historical data in the training process is reduced based on the coefficient of determination. Following this approach the execution time for training and testing process is decreased. Besides, this approach avoids using the expired information of a system in the variability of cloud federation. Experiments described in Chapter 8 will present the benefits of our algorithms.

As discussed in Chapter 4, the space of candidates in searching and optimizing are huge in cloud federation. EMO approaches have been developed based on Pareto dominance techniques and useful for non-linear programming methods which are used to solve MOPs with convex constraint functions. Among EMO approaches, Non-dominated Sorting Genetic Algorithms provide low computational complexity of non-dominated sorting. In the next chapter, we propose our approach to improve the performances of NSGAs.

Key Points

- We introduce Dynamic REgression AlgorithM, called DREAM, for estimating cost values in MOP
- We present optimization process using MOO algorithms

Publication

- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. Dynamic estimation for medical data management in a cloud federation. *International Workshop On Data Analytics solutions for Real-Life APplications (DARLI-AP@EDBT)*, Lisbon, Portugal, 2019.

NON-DOMINATED SORTING GENETIC ALGORITHM BASED ON GRID PARTITIONING

Contents

6.1 Introduction	75
6.2 NSGA-G	76
6.2.1 Main process	76
6.2.2 Non-Dominated Sorting	76
6.2.3 Filter front process	77
6.3 Discussion	80
6.3.1 Convergence	80
6.3.2 Diversity	80
6.3.3 Computation	80
6.4 Selecting the size of grid	81
6.4.1 Simple front group	81
6.4.2 Max front group	82
6.5 Conclusion	82

6.1 Introduction

Chapter 5 shows the first contribution in the estimation process for MOOP. The second step is looking for an efficient approach for searching and optimizing. Based on Non-dominated Sorting Genetic Algorithms introduced in Chapter 4, this chapter

presents the second contribution of MOOP, called Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G). We first show in Section 6.2 our algorithm based on Grid partition. Next, the quality of NSGA-G is analyzed in Section 6.3. How to select the grid partitioning is presented in the next section, Section 6.4. Finally, we conclude with Section 6.5.

6.2 NSGA-G

We propose NSGA-G to improve both diversity and convergence while having an efficient computation time by reducing the space of selected good solutions in the truncating process.

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} .

6.2.1 Main process

The process of NSGAs [39, 40] is described in Section 4.4. This section describes in more detail the main process of NSGAs. Algorithm 4 shows the steps of the processing. First, the Offspring is initialized in 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.

6.2.2 Non-Dominated Sorting

Before the truncating process, the solutions in the population with a size of $2N$ should be sorting in multiple fronts with their ranking, as shown in Algorithm 5. 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.

Algorithm 4 Main process [39, 40].

```

1: function ITERATE(Population)
2:   Offsprings  $\leftarrow \emptyset$ 
3:   while Offsprings.size < populationSize do
4:     Parent = Selection(Population)
5:     Offsprings = Offsprings  $\cup$  Evolve(Parent)
6:   end while
7:   Population = Population  $\cup$  Offsprings
8:   Population = Truncate(Population)
9:   return Population
10: end function

```

Algorithm 5 Non-dominated Sorting [39].**Require:** *R*

```

1: function SORTING(R)
2:   RinRank  $\leftarrow \emptyset$ 
3:   rank = 1
4:   remaining  $\leftarrow R$ 
5:   while RisNotEmpty do
6:     Front  $\leftarrow$  non - dominatedPopulation(remaining, rank)
7:     remaining = remaining  $\setminus$  Front
8:     RinRank = RinRank  $\cup$  Front
9:     rank ++
10:  end while
11:  return RinRank
12: end function

```

6.2.3 Filter front process

NSGA-G using Min point

NSGA-G finds the nearest smaller and bigger grid point for each solution. For example, Figure 6.1 shows an example of a two-objectives 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 $[0.35, 0.45]$, the closest smaller point is $[0.25, 0.5]$ and the nearest bigger point is $[0.5, 0.5]$.

The first strategy is avoiding to calculate 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 Figure 6.1. 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

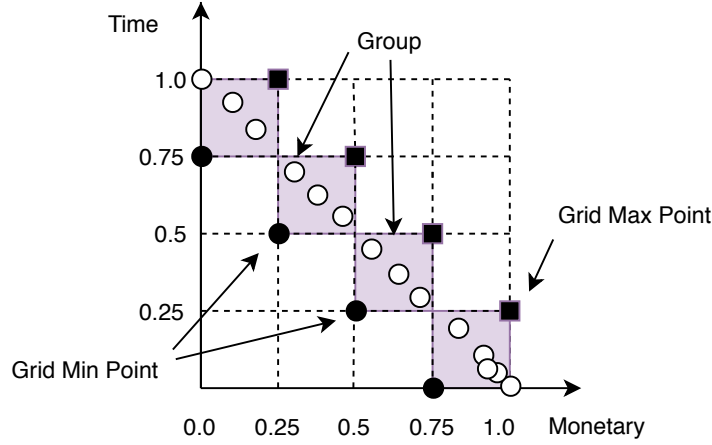


Figure 6.1 – An example of using Grid points.

Algorithm 6 Filter front in NSGA-G using Min point.

```

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  $|\mathcal{F}_l| > M$  do
8:     selectRandomGroup()
9:     removeMaxSolutionInGroup()
10:  end while
11:  return  $\mathcal{F}_l$ 
12: end function

```

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 F_3 in Fig. 6.1. In any loop, it is not necessary to compare solutions among all members in F_l , as F_3 in Figure 6.1. The second strategy is choosing randomly a group. The characteristic of diversity is maintained by this strategy. Both strategies are proposed to improve the qualities of our algorithm. Algorithm 6 shows the strategy to select $N - \sum_{j=1}^{l-1} \mathcal{F}_j$ members in \mathcal{F}_l .

The first two lines in Algorithm 6 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

Algorithm 7 Filter front in NSGA-G using Random metric.

```

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  $|\mathcal{F}_l| > M$  do
8:     selectRandomGroup()
9:     selectRandomMetric()
10:    removeWorstSolutionInGroup()
11:  end while
12:  return  $\mathcal{F}_l$ 
13: end function

```

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 metrics, Generational Distance [148], Inverted Generational Distance [30] and the Maximum Pareto Front Error [150], are used including convergence, diversity and execution time.

NSGA-G using Random metric

In MOOP, when the number of objectives is significant, any function which is used to compare solutions leads to high computation. NSGA-G using Min point uses Grid partitioning to reduce the number of solutions that needs to be compared, 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 referent 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 step in this algorithm are similar to NSGA-G using Min point, as shown from line 2 to 6. *While* loop has one more step of choosing metric randomly. *selectRandomMetric* function is used to select a natural metric among the objectives in MOP. The important characteristics of this algorithm are randomly selecting the group like NSGA-

G to keep the diversity characteristic and remove the solution among members of that group, and randomly using nature metric among various objectives to reduce the comparing time. This selection helps to avoid using an intermediate function in comparing and calculating the values of solutions.

6.3 Discussion

6.3.1 Convergence

In terms of convergence, Pareto dominance is a fundamental criterion to compare solutions. The proposed algorithm keeps the generation process of NSGAs [40, 39], except for the removed solution in the last front in preparing the population of the next generation. The convergence quality of NSGA-G is better than the original NSGAs. This will be shown in experiments of Generational Distance (GD) [148] and Inverted Generational Distance (IGD) [30] in Session 8.4.1.

6.3.2 Diversity

The proposed approach uses Grid Partitioning to guarantee that the solutions are distributed in all the solution space. In the problems of N objectives, $N \geq 4$, assuming that k solutions should be removed in the last front. In each axis coordinate with the size of grid n , the maximum number of groups in all space of N axis coordinates is n^N and we choose the number of groups in the last front including all Non-dominated solutions which should be removed is n^{N-1} .

The proposed idea is to keep the diversity characteristic of the genetic algorithm by generating k groups and removing k solutions which have the longest distance to the minimum grid point. Hence, the size of grid in the proposed algorithm is $n = \lceil k^{1/(N-1)} \rceil$, where $\lceil \cdot \rceil$ is a ceiling operator. This equation will be described in Section 6.4

6.3.3 Computation

In every generation, other NSGAs compare all solutions in the last front to remove the worst or select the best candidate. To reduce the computation of selecting a good solution in the last front, NSGA-G divides the last front into multiple groups. By dividing

into small groups, the proposed algorithm selects a good solution in a small group, that helps to accelerate the selection process in comparison with other approaches scanning all solutions of the last front.

6.4 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.

6.4.1 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

$$n = \lceil k^{\frac{1}{N-1}} \rceil. \quad (6.1)$$

For example, Figure 6.2 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 $n = \lceil k^{\frac{1}{N-1}} \rceil = 4$.

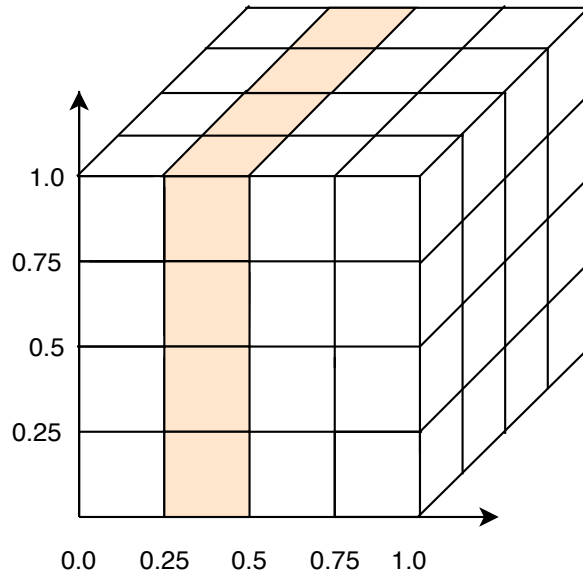


Figure 6.2 – A simple front group.

6.4.2 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 . The *front group* which has the largest number of groups includes N planes. Hence, the number of groups in this *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

$$n^N - (n - 1)^N = k. \quad (6.2)$$

For instance, Figure 6.3 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 . A max *front group* includes $4^3 - 3^3 = 64 - 27 = 37$.

6.5 Conclusion

In this chapter, we have presented our contribution to EMO algorithms, a Non-dominated Sorting Genetic Algorithm based on Grid partitioning in NSGAs class. Our approach can be applied to MOOPs. This chapter also discusses the advantages of our algorithm compared to original NSGAs. The characteristic of NSGA-G promises to

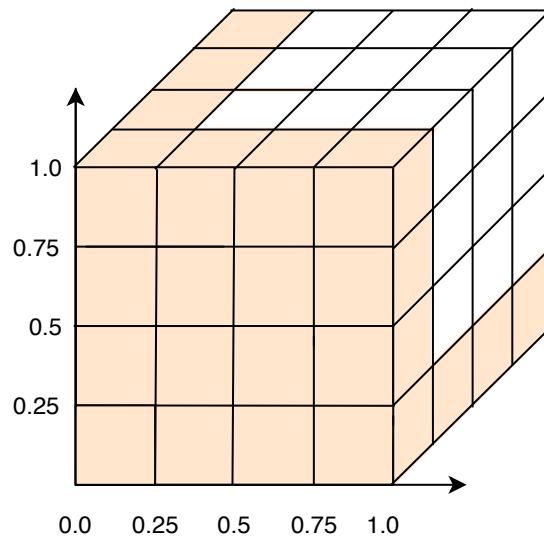


Figure 6.3 – A max front group.

improve convergence, diversity and execution time in MOOPs. Besides, the way to determine the grid point is presented. This method is adapted to the number of candidate solutions in the last front of NSGA process.

As discussed in Chapter 2, a medical data set, such as DICOM, needs to store in hybrid data storage to improve query processing while saving storage space. In the next chapter, we detail our proposals for the optimization of hybrid data storage configuration in a cloud federation.

Key Points

- We propose Non-dominated Sorting Genetic Algorithm based on Grid Partitioning with Min point and Random metric.
- We present two methods of choosing the grid partitioning, including the simple and max front group.

Publication

- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. An efficient multi-objective genetic algorithm for cloud computing: NSGA-G. *International workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD@BigData)*, Seattle, WA, USA, 2018.

HYBRID DATA STORAGE CONFIGURATION IN CLOUD FEDERATION

Contents

7.1 Introduction	85
7.2 Medical system on cloud federation	86
7.2.1 MIDAS	86
7.2.2 IRES	86
7.2.3 Hybrid data storage configuration	87
7.2.4 Validation	88
7.3 Hybrid data storage configuration	90
7.3.1 Two phases of generating data storage configuration	90
7.3.2 Implementation	95
7.4 Optimizing data storage configuration	104
7.4.1 Finding Pareto configuration set	104
7.4.2 Finding the best configuration	104
7.5 Conclusion	105

7.1 Introduction

Chapter 5 and 6 introduce two algorithms of estimation accurate cost value, searching and optimizing for Multi-Objective Optimization Problems in a cloud federation. This chapter introduces the detail of a hybrid data storage configuration of the medical system in cloud environment and how to find a good hybrid data storage configuration following the required quality of workload [97]. Section 7.2 shows an overview of Medical Data Management System for a cloud federation and the background of our approach

in data storage configuration. Section 7.3 shows the hybrid data storage configuration implement following the automatic generation approach [97]. After that, the hybrid data storage configuration optimized by our solution (NSGA-G) is presented in Section 7.4. Finally, we conclude with Section 7.5.

7.2 Medical system on cloud federation

7.2.1 MIDAS

We propose Medical Data Management System (MIDAS), a DICOM management system for cloud federation. MIDAS aims to provide both efficient DICOM management system based on hybrid row and column layouts and query processing strategies to integrate existing information systems (with their associated cloud provider and data management system) for clinics and hospitals. Figure 7.1 presents an example of MIDAS. IReS optimizes workflows between different data sources and queries on graph data combining with related data on different clouds such as Amazon Web Services, Microsoft Azure and Google Cloud Platform. IReS takes advantage of multi-engine and data stores on clouds. Meanwhile, HYTORMO generates a hybrid data storage configuration on clouds using an automatic approach based on the workload related to DICOM data at the initialization process. The hybrid store takes advantage of both traditional storage techniques, sequence files and Record Columnar File in Hive [129] or PostgreSQL [133] and takes into account various kind of queries, including OLAP and OLTP.

7.2.2 IRES

IReS provides a method of optimizing cost-based workflow and customizable resource management of diverse execution and various storage engines. IReS receives information on data, operators and user-defined policy. Then, it optimizes a workflow respecting user policy and puts the optimal plan into the physical infrastructure. Modelling module, as shown in Figure 7.1, predicts the execution time by the model which is chosen by comparing many machine learning algorithms. The module tries to build models using the total information for training and testing process.

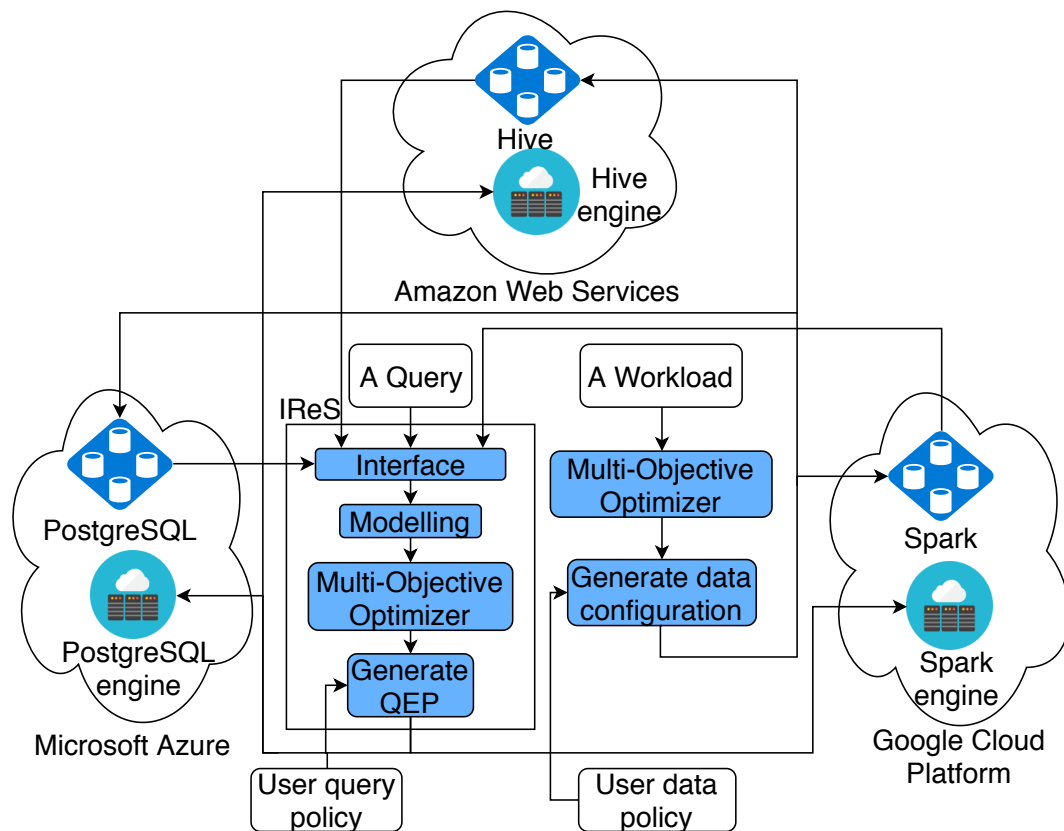


Figure 7.1 – An example of MIDAS system

7.2.3 Hybrid data storage configuration

To optimize the performances for both OLAP and OLTP workloads, the hybrid store has two strategies to optimize storage and query.

Data Storage Strategy

Data storage strategy aims to optimize query performance and storage space over a mixed OLTP and OLAP workload by extracting, organizing and storing data to reduce space, tuple construction and I/O cost. In this strategy, all tables will be decomposed into multiple sub-tables and stored in row or column stores of the hybrid store. For instance, the entity *Patient* table in DICOM can be vertically partitioned into two tables called *RowPatient* and *ColumnPatient*. A group of attributes classified as frequently-accessed-together attributes can be stored in a row table, e.g. *RowPatient*. Meanwhile, other groups are classified as optional attributes and stored in a column store, e.g.

ColumnPatient. Each attribute belongs to one group except that it is used to join the tables together. This strategy removes the null rows. Concretely, when a query requests for attributes from many sub-tables, the hybrid store will change data storage configuration to have efficient query processing in joining operators between sub-tables.

Query Processing Strategy

In order to improve the performance of query processing in a cloud environment, the hybrid store needs to modify sub-tables to reduce the left-outer joins and irrelevant tuples in the input tables of the join operation. When a query needs attributes from many sub-tables, the hybrid store will change data storage configuration to have an efficient query processing in joining operators between sub-tables. The query performance is negatively impacted if the query execution needs attributes by joining many tables. Therefore, the hybrid store needs to reconstruct result tuples and the storage space will be increased to store surrogate attributes.

Denoting $\mathbf{W} = (A, Q, AUM, F)$ as a workload related to table \mathbf{T} [97], it comprises four elements including query set Q in workload \mathbf{W} , attribute set A of table \mathbf{T} , attribute usage matrix AUM of \mathbf{T} and frequencies set of queries F in workload \mathbf{W} . The hybrid data storage configuration is formed by four parameters including the weight of similarity α , clustering threshold β , merging threshold θ and data layout threshold λ . Depending on these four parameters and workload \mathbf{W} , HYTORMO automatically creates a data storage configuration for the hybrid store.

7.2.4 Validation

Modelling module of IReS requires low computation cost and accurate method to estimate the cost values. As mentioned in Chapter 5, machine learning methods can lead to use expired information. Hence, our proposed method, DREAM, is integrated into the Modelling module to predict the cost values with low computation cost in a cloud environment. Moreover, the Multi-Objective Optimizer requires an efficient algorithm to find an approximation of a Pareto-optimal solution. However, the space of candidate solutions in MOOP for DICOM hybrid data is huge [97] and generating the Pareto-optimal front is often infeasible due to high complexity [160].

For example, the automatic approach generating hybrid DICOM data storage configuration in [97], called HYTORMO, using 4 predefined parameters the weight of sim-

Table 7.1 – AUM and frequencies

	a_1	a_2	a_3	a_4	a_5	a_6	F
q_1	0	1	1	1	1	0	600
q_2	0	0	1	1	1	1	100
q_3	0	0	0	1	1	1	700
q_4	1	1	0	0	0	0	1000
q_5	1	1	1	0	0	0	200
q_6	0	1	1	0	0	0	400

UID	a1	a2	a3	a4	a5	a6
01						
02						
03						
04						
05						
06						
07						
08						
09						
10						

: Null value
 : Non-null value

Figure 7.2 – The horizontal table T .

ilarity α , clustering threshold β , merging threshold θ and data layout threshold λ to generate configuration for a given workload and data specific information, as shown in Table 7.1 and Figure 7.2. The results of automatic approach in HYTORMO [97] are shown in Table 7.2. Besides of these configuration, there are many other ones with different values of $\alpha, \beta, \theta, \lambda$.

The vast space of data storage configuration candidates in a hybrid storage system leverages an alternative solution to find a Pareto-optimal one. Our method applies NSGA-G, an efficient MOO algorithm, to the Multi-Objective Optimizer to find an approximation of a Pareto-optimal solution. Our proposals are to improve the accuracy of cost value prediction with low computation cost and to solve MOOP in both the querying and storing configurations with an efficient algorithm in a cloud environment. Hence, hybrid DICOM data storage configuration should be optimized by MOO algorithm, such as NSGA-G.

Table 7.2 – AUM and frequencies

	α	β	θ	λ	Configuration
G_1	0	0	0	0	$[a_1, a_2, a_3, a_4, a_5, a_6]$ =row-store
G_2	0	0.4	0	0	$[a_1, a_2, a_3, a_4, a_5, a_6]$ =row-store
G_3	0.5	0.4	0	0	$[a_1, a_2, a_3, a_4, a_5, a_6]$ =row-store
G_4	0.5	0.4	0	0.3	$[a_1, a_2, a_3, a_4, a_5, a_6]$ =column-store
G_5	0.5	0.4	0.2	0.3	$[a_1, a_2, a_3, a_4, a_5, a_6]$ =column-store
G_6	0.5	0.8	0.2	0.8	$[a_1, a_2]$ =column-store, $[a_3, a_4, a_5, a_6]$ =column-store
G_7	0	0.5	0.2	0.7	$[a_1, a_2, a_3]$ =column-store, $[a_4, a_5, a_6]$ =row-store

7.3 Hybrid data storage configuration

7.3.1 Two phases of generating data storage configuration

Following the automated design framework [97] based on workload and data-specific information to produce data storage configurations for DICOM data. The method is generating a candidate configuration using as inputs:

- Workload-specific inputs including Attribute Usage Matrix (AUM) and F (query frequencies),
- Data-specific input including the horizontal table T ,
- Parameter including
 - α (weight of similar),
 - β (clustering threshold),
 - θ (merging threshold),
 - λ (data layout threshold).

The method performs two phases: clustering and merging-selection. The first phase aims to decrease storage space and reduce irrelevant attribute accesses. The second phase is used to improve both reconstruction cost and the number of irrelevant accesses.

Clustering Phase

The first phase is used to analyze both workload and data specific information on the quality of vertical partitioning result. From the given table T and workload W , HY-TORMO computes two similarity measures *Attribute Access Similarity* and *Attribute Density Similarity* between every pair of attributes. The first measure analyses the

workload information. It is computed by the *Attribute Usage Matrix* and query frequencies F and presented by *Attribute Access Similarity Matrix*. The *Attribute Access Similarity* between two attribute a_x and a_y is computed by the Jaccard Coefficient [93, 85]

$$\text{AttributeAccessSimilarity}(a_x, a_y) = \frac{\sum_{i=1}^m [(AUM[i][a_x] \wedge AUM[i][a_y]) \times f_i]}{\sum_{i=1}^m [AUM[i][a_x] \times f_i] - \sum_{i=1}^m [(AUM[i][a_x] \wedge AUM[i][a_y]) \times f_i] + \sum_{i=1}^m [AUM[i][a_y] \times f_i]}, \quad (7.1)$$

where \wedge is a binary bitwise AND operator and m is the number of queries in workload W . The Algorithm 8 shows the way to calculate the *Attribute Access Similarity Matrix*.

Algorithm 8 Attribute Access Similarity.

Require: AUM , $frequencies$

- 1: $AUM(m \times n)$: Attribute Usage Matrix including m row and n attributes
- 2: F : Query frequencies

Ensure: AAS

- 3: AAS : Attribute Access Similarity ($n \times n$)
 - 4: **function** ATTRIBUTEACCESSSIMILARITY(AUM, F)
 - 5: **for** $i \leftarrow 0$ to $AUM[0].length$ **do**
 - 6: **for** $j \leftarrow 0$ to $AUM[0].length$ **do**
 - 7: **if** $AUM[i][j]=1$ **then** $AUMbit[i][j] = \text{true}$;
 - 8: **else**
 - 9: $AUMbit[i][j] = \text{false}$;
 - 10: **end if**
 - 11: **end for**
 - 12: **end for**
 - 13: $AAS = \emptyset$
 - 14: **for** $i \leftarrow 0$ to $AUM[0].length$ **do**
 - 15: $A_i = \text{takeColumnBitMatrix}(AUMbit, i)$
 - 16: **for** $j \leftarrow 0$ to $AUM[0].length$ **do**
 - 17: $A_j = \text{takeColumnBitMatrix}(AUMbit, j)$
 - 18: $\text{and} = \text{andLogic}(A_i, A_j)$
 - 19: $\text{sumAndLogic} = \text{sumFrequencies}(\text{and}, F)$
 - 20: $AAS[i][j] = \frac{\text{sumAndLogic}}{\text{sumFrequencies}(A_i, F) + \text{sumFrequencies}(A_j, F) - \text{sumAndLogic}}$
 - 21: **end for**
 - 22: **end for**
 - 23: **return** AAS
 - 24: **end function**
-

First, the algorithm finds the attributes which are used together by generating $AUMbit[i][j]$

matrix, from line 5 to 12. After that *Attribute Access Similarity Matrix* showing the relationship between two attributes are computed by 7.3.1, line 20.

The second one captures the data information, and shows in *Attribute Density Similarity Matrix*. The matrix is computed by the information in the given horizontal table T . The measure is also calculated by the Jaccard Coefficient as follows:

$$AttributeDensitySimilarity(a_x, a_y) = \frac{\sum_{i=1}^{|T|} \{isNotNull(T[i][a_x]) \wedge isNotNull(T[i][a_y])\}}{\sum_{i=1}^{|T|} isNotNull(T[i][a_x]) + \sum_{i=1}^{|T|} isNotNull(T[i][a_y]) - \sum_{i=1}^{|T|} \{isNotNull(T[i][a_x]) \wedge isNotNull(T[i][a_y])\}}, \quad (7.2)$$

and the Algorithm 9 presents how to calculate *Attribute Density Similarity Matrix*.

Algorithm 9 Attribute Density Similarity.

Require: T

- 1: $T(m \times n)$: Horizontal table including m tuples and n attributes

Ensure: ADS

- 2: ADS : Attribute Density Similarity ($n \times n$)
 - 3: **function** ATTRIBUTEDENSITYSIMILARITY(T)
 - 4: $isNotNull = isNotNull(T)$ $ADS = \emptyset$
 - 5: **for** $i \leftarrow 0$ to $T[0].length$ **do**
 - 6: $A_i = takeColumnBitMatrix(isNotNull, i)$
 - 7: **for** $j \leftarrow 0$ to $T[0].length$ **do**
 - 8: $A_j = takeColumnBitMatrix(isNotNull, j)$;
 - 9: $and = andLogic(A_i, A_j)$;
 - 10: $andLogic = checkSumVector(and)$;
 - 11: $ADS[i][j] = \frac{andLogic}{AASM.checkSumVector(A_i) + AASM.checkSumVector(A_j) - andLogic}$
 - 12: **end for**
 - 13: **end for**
 - 14: **return** ADS
 - 15: **end function**
-

The *Hybrid Similarity* between each pair of attributes, presented by *Hybrid Similarity Matrix*, is calculated by the *Attribute Access Similarity* and *Attribute Density Similarity* with the weight of similar, α , as follows:

$$HybridSimilarity(a_x, a_y) = \alpha \times AttributeAccessSimilarity(a_x, a_y) + (1 - \alpha) \times AttributeDensitySimilarity(a_x, a_y), \quad (7.3)$$

where α is the predefined weight parameter, such as $\alpha \in [0, 1]$. The *Hybrid Similarity Matrix* can be calculated by AASM and ADSM, as follows:

$$HSM = \alpha \times AASM + (1 - \alpha) \times ADSM \quad (7.4)$$

Algorithm 10 HybridSimilaritySimilarity [97].

Require: AAS, ADS, α

1: AAS: Attribute Access Similarity ($n \times n$)

2: ADS: Attribute Density Similarity ($n \times n$)

3: α : Weight of similar

Ensure: HS

4: HS: Hybrid Similarity($n \times n$)

5: **function** HYBRIDSIMILARITY(AAS, ADS, α)

6: **for** $i \leftarrow 0$ to $AAS.length$ **do**

7: **for** $j \leftarrow 0$ to $AAS[0].length$ **do**

8: $HS[i][j] = \alpha * AAS[i][j] + (1 - \alpha) * ADS[i][j]$

9: **end for**

10: **end for**

11: **return** HS

12: **end function**

Finally, a set of resulting is a column groups $C_{in} = \{C_{i,1}, C_{i,2}, \dots, C_{i,z}\}$, is shown as *setCluster* in Algorithm 16.

Merging-Selection Phase

To improve the query performance, the second phase tries to reduce both the tuple reconstruction cost and the number of irrelevant attribute accesses. The hybrid data storage configuration should take care of efficient storage in terms of workload and query processing. The input of this phase is the resulting column groups in the clustering phase. *Intra-Cluster Access Similarity* and *Inter-Cluster Access Similarity* [126, 97] are used in this phase to reduce the number of joins and apply a suitable layout to each column group.

The *Intra-Cluster Access Similarity* of a single cluster $C_{i,u}$ of a data storage configuration G_i is calculated as follows:

$$\begin{aligned}
 \text{IntraClusterAccessSimilarity}(C_{i,u}) = & \\
 & \begin{cases} \frac{\sum_{a_x \in C_{i,u}, a_y \in C_{i,u}, x \neq y} \text{AttributeAccessSimilarity}(a_x, a_y)}{|C_{i,u}| \times (|C_{i,u}| - 1)}, & \text{if } |C_{i,u}| > 1 \\ 1, & |C_{i,u}| = 1, \end{cases} \quad (7.5)
 \end{aligned}$$

where $C_{i,u}$ is a component of a set of column groups $C_i = \{C_{i,1}, C_{i,2}, \dots, C_{i,z}\}$ in a candidate data storage configuration $G_i = (C_i, L_i)$, and $L_i = \{L_{d_1}(C_{i,1}), L_{d_2}(C_{i,2}), \dots, L_{d_z}(C_{i,z})\}$ is a set of data layouts for the column groups. $L_{d_x}(C_{i,x})$ shows the layout d_x of a column group $C_{i,x}$, where the row-store is denoted by $d_x = 1$ and the column-store is showed by $d_x = 0$, or vice versa.

Algorithm 11 Intra-Cluster Access Similarity [97].

Require: AAS, Cu

1: $AAS(m \times n)$: Attribute Access Similarity ($n \times n$)

2: Cu : Cluster

Ensure: $\text{IntraClusterAccessSimilarity}$

3: **function** INTRACLUSTERACCESSSIMILARITY(AAS, Cu)

4: **if** $|Cu| = 1$ **then** return 1

5: **else**

6: **for** $u:Cu$ **do**

7: **for** $v:Cu$ **do**

8: **if** $u \neq v$ **then**

9: $sum = sum + AAS[u][v]$

10: **end if**

11: **end for**

12: **end for**

13: **return** $\frac{sum}{(|Cu|) * (|Cu| - 1)}$

14: **end if**

15: **end function**

The Inter-Cluster Access Similarity between cluster $C_{i,u}$ and $C_{i,v}$ of a data storage configuration G_i is defined as follows

$$\begin{aligned}
 \text{InterClusterAccessSimilarity}(C_{i,u}, C_{i,v}) = & \\
 & \left\{ \frac{\sum_{a_x \in C_{i,u}, a_y \in C_{i,v}} \text{AttributeAccessSimilarity}(a_x, a_y)}{|C_{i,u}| \times |C_{i,v}|}, \quad (7.6)
 \end{aligned}$$

where $u \neq v$.

Algorithm 12 Inter-Cluster Access Similarity [97].

Require: AAS, C_u, C_v

1: $AAS(m \times n)$: Attribute Access Similarity ($n \times n$)

2: C_u, C_v : Cluster

Ensure: *InterClusterAccessSimilarity*

3: **function** INTERCLUSTERACCESSSIMILARITY(AAS, C_u, C_v)

4: $sum = 0$

5: **for** $u:C_u$ **do**

6: **for** $v:C_v$ **do**

7: $sum = sum + AAS[u][v]$

8: **end for**

9: **end for**

10: **return** $\frac{sum}{(|C_u| * |C_v|)}$

11: **end function**

Algorithm 12 shows the way to calculate *Inter-Cluster Access Similarity* between two clusters.

The merging-selecting phase groups a pair of columns together to create a new group if their *Inter-Cluster Access Similarity* is bigger than or equal to the threshold θ . A column group is stored in a row store if the *Intra-Cluster Access Similarity* is greater than or equal to the threshold λ ; otherwise, it is stored in a column store. The output of this phase is a candidate of data storage configuration $G_i = (C_i, L_i)$.

7.3.2 Implementation

The automatic approach applies two phases, *Clustering* and *Merging-selecting* phase, to generate a candidate storage configuration. The algorithms are based on the information of a given table, workloads and four parameters, such as weight of similar α , clustering threshold β , merging threshold θ , data layout threshold λ .

Generating a candidate storage configuration

Algorithm 13 presents the approach to generate automatically a hybrid data storage configuration, including two phases. In the clustering phase, *Attribute Access Similarity* matrix is created to present the relationship between every pair of attributes in the given workload and table. *Attribute Density Similarity* (ADS) matrix is generated to show the

relationship between every pair of attributes in terms of the density. After that, *Hybrid Similarity*(HS) matrix is constructed based on AAS, ADS, and the weight of similarity α . *Attribute Access Correlation* matrix is built to describe how many times two attributes are simultaneously accessed. The number of times two attributes simultaneously have non-null values are described in the *Attribute Density Correlation* (ADC) matrix. After that, the result clusters is presented as a set of clusters $C = \{C_1, C_2, \dots, C_z\}$. At the end of this phase, the clustering threshold, β ranges from 0 to 1, is used to add unclustered attributes, a_u , to a cluster C_i when Hybrid Similarity between a_u and every attributes in C_i is not less than the given β .

The merging-selecting phase calls a function, *MergeAndSelectStore*, depending on data layout threshold λ and merging threshold θ . The procedure checks every pair of clusters and merging threshold to determine which pair of clusters are merged into a new one. After that, the procedure decides which data layout is used to store attributes in a cluster based on data layout threshold λ . The result of merging-selecting phase is a candidate data storage configuration $G = (C, L)$, where C is a set of clusters and L is a set of suggested data layouts.

This approach concerns all the information of data and workload related to given table (i.e., Attribute Usage Matrix, *AUM*, query frequencies, F , horizontal table T). Besides, a new candidate data storage configuration G of a table is generated corresponding parameters, $\alpha, \beta, \lambda, \theta$. Each configuration has the statistics, such as the storage cost, Null-ratio, the total number of joins, the total number of scanned data cells, monetary storage cost, etc. There is no solution which have all objective better than other one. Hence, we should optimize the hybrid DICOM data storage configuration problem by MOOP.

Attribute Access Correlation

The Algorithm 14 is used to generate *Attribute Access Correlation* matrix ($n \times n$). This matrix shows the correlation between two attributes in given table T . It presents how many times two attributes are accessed together. First, *Attribute Access Correlation* matrix, AAC, is initialized with n rows and n columns. After that, the algorithm checks every row of AUM. For each row in AUM, the array *row* store the value of current row. Next, the number of times in which two attributes k and l are access together is increased by $AAC[k][l] = AAC[k][l] + row[l] * frequencies[i]$. The result of this algorithm is AAC matrix, as shown in Algorithm 14.

Algorithm 13 Generating Storage Configuration [97].

Require: $AUM, T, F, \alpha, \beta, \lambda, \theta$

- 1: $AUM(m \times n)$: Attribute Usage Matrix including m rows and n attributes
- 2: $T(m \times n)$: Horizontal table including m tuples and n attributes
- 3: F : Query frequencies
- 4: α : Weight of similarity
- 5: β : Clustering threshold
- 6: λ : Data layout threshold
- 7: θ : Merging threshold

Ensure: *InterClusterAccessSimilarity*

- 8: **function** GENERATESTORAGECONFIGURATION($AUM, T, F, \alpha, \beta, \lambda, \theta$)
 - 9: $AAS = AttributeAccessSimilarity(AUM, F)$
 - 10: $ADS = AttributeDensitySimilarity(T)$
 - 11: $HS = HybridSimilarity(AAS, ADS)$
 - 12: $AAC = AttributeAccessCorrelation(AUM, F)$
 - 13: $ADC = AttributeDensityCorrelation(T)$
 - 14: // Clustering phase
 - 15: $C = ClusterAttributes(AAC, ADC, HS, \alpha, \beta)$
 - 16: // Merging-selecting phase
 - 17: $G = MergeAndSelectStore(AAS, C, \lambda, \theta)$
 - 18: **return** G
 - 19: **end function**
-

Algorithm 14 Attribute Access Correlation [97].

Require: AUM , $frequencies$

- 1: $AUM(m \times n)$: Attribute Usage Matrix including m row and n attributes
- 2: $frequencies$: Query frequencies

Ensure: AAC

- 3: AAC : Attribute Access Correlation ($n \times n$)
- 4: **function** ATTRIBUTEACCESSCORRELATION(AUM, F)
- 5: **for** $i \leftarrow 0$ to $AUM[0].length$ **do**
- 6: **for** $j \leftarrow 0$ to $AUM[0].length$ **do**
- 7: $AAC[i][j] = 0$;
- 8: **end for**
- 9: **end for**
- 10: **for** $i \leftarrow 0$ to $AUM.length$ **do**
- 11: $row = \emptyset$
- 12: **for** $j \leftarrow 0$ to $AUM[0].length$ **do**
- 13: **if** $AUM[i][j] == 1$ **then**
- 14: $row[j] = 1$
- 15: **else**
- 16: $row[j] = 0$
- 17: **end if**
- 18: **end for**
- 19: **for** $k \leftarrow 0$ to $AUM[0].length$ **do**
- 20: **if** $row[k] == 1$ **then**
- 21: **for** $l \leftarrow k$ to $AUM[0].length$ **do**
- 22: $AAC[k][l] = AAC[k][l] + row[l] * frequencies[i]$
- 23: **end for**
- 24: **end if**
- 25: **end for**
- 26: **end for**
- 27: **return** AAC
- 28: **end function**

Attribute Density Correlation

Algorithm 15 is used to generate *Attribute Density Correlation* matrix ($n \times n$). This matrix shows the correlation between two attributes in the Table T . It presents how many times two attributes having non-null values and are accessed together. First, *Attribute Density Correlation* matrix, ADC , is initialized with n rows and n columns. After that, the algorithm checks every row of T . For each row in T , the array row store the value of current row. Next, the number of times in which two attributes k and l having non-null values are access together is increased by $ADC[k][l] = ADC[k][l] + row[l]$. The result of this algorithm is AAC matrix, as shown in Algorithm 15.

Clustering Attribute

Algorithm 16 is used to generate the clusters in the clustering phase. The impact of both workload and data information is considered in this algorithm. The approach aims to reduce storage space and improve workload performance simultaneously. The attributes of given table T is divided into a set of clusters $setCluster$. The inputs of this algorithm are AAC , ADC , HS and parameters, i.e., the weight of similarity α , clustering threshold β . First, AAC or ADC is selected depending on α values. For example, if $\alpha \geq 0.5$, the approach chooses AAC . Otherwise, ADC is considered. Next, a new cluster is added into $setCluster$ and all the attributes having the same condition of Hybrid Similarity value. The procedure is repeated until the left attribute is empty.

The clusters in $setCluster$ is arranged in the decreasing order of the important attributes level. The more attribute access frequency or data density (non-null values) is selected before the less ones. The first step is selecting the frequency attribute or data density. Depending on the parameter α , the method chooses AAC or ADC . After creating empty $setCluster$, the most important attribute is added to a new cluster (i.e., $cluster_{index}$). After that, each left attribute aL is compared to each attribute aC in a cluster in terms of Hybrid Similarity value. If the value is less than the clustering threshold β , aL is added to $cluster_{index}$ in which includes aC . The repeat loop is stopped when the left attributes are empty. Finally, the output of the algorithm is a set of clusters, $setCluster$.

The parameter of the clustering threshold is predefined by users based on experiments. If the value of β is small, many attributes having the same range value of the Hybrid Similarity are added into a cluster. Hence, the number of clusters is small, but

Algorithm 15 Attribute Density Correlation [97].

Require: T

1: $T(m \times n)$: Horizontal table including m tuples and n attributes

Ensure: ADC

2: ADC : Attribute Density Correlation ($n \times n$)

3: **function** ATTRIBUTE DENSITY CORRELATION(T)

4: **for** $i \leftarrow 0$ to $T[0].length$ **do**

5: **for** $j \leftarrow 0$ to $T[0].length$ **do**

6: $ADC[i][j] = 0$;

7: **end for**

8: **end for**

9: **for** $i \leftarrow 0$ to $T.length$ **do**

10: $row = \emptyset$

11: **for** $j \leftarrow 0$ to $T[0].length$ **do**

12: **if** $T[i][j] \neq null$ **then**

13: $row[j] = 1$

14: **else**

15: $row[j] = 0$

16: **end if**

17: **end for**

18: **for** $k \leftarrow 0$ to $T[0].length$ **do**

19: **if** $row[k] == 1$ **then**

20: **for** $l \leftarrow k$ to $T[0].length$ **do**

21: $ADC[k][l] = ADC[k][l] + row[l]$

22: **end for**

23: **end if**

24: **end for**

25: **end for**

26: **return** ADC

27: **end function**

the size of the clusters is large. As a consequence, wide tables with a large number of null values or irrelevant attributes are created. In contrast, if β is large, less attributes having high hybrid similarity are added into a cluster. The result is a large number of clusters including a small number of attributes. The number of null values is reduced, but the expensive join operators are generated in the reconstructing result tuples across narrow tables.

Merging and Selecting Stores

Algorithm 17 implements the second phase of the generating hybrid data storage configuration process. It is used to improve query performance. The result of the first phase is a set of clusters, including attributes (non-overlapping) of the given table T . When many attributes of a tuple are accessed, the query performance may be reduced by the reconstructing tuple process. The second phase is created to reduce both the tuple reconstruction cost and the number of irrelevant attribute accesses.

In the first step, the algorithm focuses on reducing the tuple reconstruction cost, which impacts query performance. Two clusters are grouped together if the *Inter Cluster Access Similarity* of them is greater than or equal to a given merging threshold θ . This parameter is predefined at the beginning of the generating data storage configuration process. The algorithm checks the *Inter Cluster Access Similarity* value between two clusters. If the condition is satisfied, being bigger than or equal to merging threshold θ , two clusters are merged together. Otherwise, they are two independent clusters. At the end of this step, the list of clusters is generated.

In the second step, the algorithm focuses on reducing the number of irrelevant attribute accesses. The data layout (row or column) has an impact on the query performance when the workload accesses irrelevant attribute. If the attributes in the same cluster are frequently accessed together, they need to be stored in row-oriented data layout storage. Otherwise, they should be stored in column-oriented data layout storage. The value of the *Intra Cluster Access Similarity* is used in this step. The algorithm examines the *Intra Cluster Access Similarity* of a cluster. If the value is larger than the data layout threshold λ , attributes in this cluster is stored in row-oriented data layout storage, and vice versa. At the end of this step, the list of suggested data layout $layout_i$ is created and stored along with C_i in *configuration*.

Algorithm 16 ClusterAttributes [97].

Require: AAC, ADC, HS, α

- 1: AAC: Attribute Access Correlation ($n \times n$)
- 2: ADC: Attribute Density Correlation ($n \times n$)
- 3: HS: Hybrid Similarity ($n \times n$)
- 4: α : Weight of similar
- 5: β : Clustering threshold

Ensure: $setCluster$

- 6: $setCluster$: a set of resulting cluster (column groups)
- 7: **function** CLUSTERATTRIBUTES($AAC, ADC, HS, \alpha, \beta$)
- 8: $Matrix = \emptyset$
- 9: **if** $\alpha \geq 0.5$ **then**
- 10: $Matrix = AAC$
- 11: **else**
- 12: $Matrix = ADC$
- 13: **end if**
- 14: $setCluster = \emptyset, index = 0, cluster_{index} = \emptyset$
- 15: $IndexAttribute = IndexAttribute(Matrix)$
- 16: **while** $|Attribute| > 0$ **do**
- 17: $IndexMax = 0, MaxValue = 0$
- 18: **for** $index:IndexAttribute$ **do**
- 19: **if** $Matrix[index][index] > MaxValue$ **then**
- 20: $MaxValue = Matrix[index][index];$
- 21: $IndexMax = index;$
- 22: **end if**
- 23: **end for**
- 24: $cluster_{index} = cluster_{index} \cup \{IndexMax\}$
- 25: $Attribute.remove(IndexMax)$
- 26: **for** $aL : Attribute$ **do**
- 27: $similarity = true$
- 28: **for** $aC : cluster_{index}$ **do**
- 29: **if** $(aC < aL \text{ and } HS[aC][aL] < \beta)$ **or** $(aC > aL \text{ and } HS[aC][aL] > \beta)$
- 30: $similarity = false$
- 31: **break**
- 32: **end if**
- 33: **end for**
- 34: **if** $similarity$ **then**
- 35: $cluster_{index} = cluster_{index} \cup \{aL\}$
- 36: **end if**
- 37: **end for**
- 38: $setCluster = setCluster \cup cluster_{index}$
- 39: $index++$
- 40: **end while**
- 41: **return** $setCluster$
- 42: **end function**

Algorithm 17 Merging And Selecting Stores [97].**Require:** $AAS, Cluster, \lambda, \theta$

- 1: $AAS(m \times n)$: Attribute Access Similarity ($n \times n$)
- 2: C_{in} : Cluster
- 3: λ : Data layout threshold
- 4: θ : Merging threshold

Ensure: *configuration*

```

5: function MERGEANDSELECTSTORE( $AAS, C_{in}, \lambda, \theta$ )
6:    $C = C_{in}$ 
7:    $configuration = \emptyset$ 
8:    $found = true$ 
9:   // Merge two clusters together depending on Inter-Cluster Access Similarity
10:  while  $found$  do
11:     $MaxSim = 0, found = false, indexU = indexV = 0$ 
12:    for  $i \leftarrow 0$  to  $|C_{in}| - 1$  do
13:      for  $j \leftarrow i + 1$  to  $|C_{in}|$  do
14:         $currentSim = interClusterAccessSimilar(C_{in}(i), C_{in}(j), ASM)$ 
15:        if  $currentSim \geq \theta$  and  $currentSim > MaxSim$  then
16:           $MaxSim = currentSim, found = true, indexU = i, indexV = j$ 
17:        end if
18:      end for
19:    end for
20:    if  $found$  then
21:       $C = C \setminus C(indexU)$ 
22:       $C = C \setminus C(indexV)$ 
23:       $C = C \cup Merge(C(indexU), C(indexV))$ 
24:    end if
25:  end while
26:  // Selecting a row/column layout for each cluster depending on Inter-Cluster
  Access Similarity
27:   $store = \emptyset$ 
28:  for  $i \leftarrow 0$  to  $|C|$  do
29:    if  $intraClusterAccessSimilar(C(i), ASM) \geq \lambda$  then
30:       $store.add(row)$ 
31:    else
32:       $store.add(column)$ 
33:    end if
34:  end for
35:  for  $i \leftarrow 0$  to  $|C|$  do
36:     $configuration.put(C(i), store.layout(i))$ 
37:  end for
38:  return  $configuration$ 
39: end function

```

7.4 Optimizing data storage configuration

7.4.1 Finding Pareto configuration set

In order to reduce storage space and improve workload execution performance, the automatic generation of a hybrid data storage configuration is presented above [97]. The method generates a data storage configuration based on four parameters. Each parameter is a real value and belongs to a range of $[0, 1]$. They are predefined by users based on experiments. Hence, the authors can not give an optimal data storage configuration of a hybrid system. They state that the space of parameter and data storage configuration is huge.

This section introduces an algorithm to find an optimal data storage configuration using NSGA-G, as shown in Algorithm 18. First, the hybrid data storage configuration is formed by four parameters including the weight of similarity α , clustering threshold β , merging threshold θ and data layout threshold λ . Depending on these four parameters, HYTORMO automatically creates a data storage configuration of a hybrid store. However, the authors did not optimize the space of solutions of data storage configuration. Hence, in the space of four parameters in $[0, 1]$, we use NSGA-G to look for a Pareto set of data storage configuration. Algorithm 18 finds the best data storage configuration for a table **T**. Line 10 generates a Pareto set of data storage configuration. After that, Line 12 uses Algorithm 3 to return the best solution in this set with the weight sum model **S** and the constraint **B** [67], as shown in Algorithm 19.

7.4.2 Finding the best configuration

Among Pareto data storage configuration set, G , the best solution should be chosen. Function *BestInPareto* is used to choose the best candidate in the set of non-dominated candidates, called Pareto set. At the final step of MOOP, we have a Pareto set of candidates. Depending on the user preferences, we have different weighted sum model **S** and constraints **B**. At the beginning, all candidates which have smaller cost function values than the value of constraints are added to the new set of non-dominated candidates, as shown in line 2. If the size of this set is not empty, line 3, the scalar value of each candidate is determined by the weighted sum model **S** and the best solution is selected by the minimum value of $C(g)$, as shown in Line 4. In contrast, when the size of new non-dominated candidates set is zero, the best candidate is chosen in the orig-

Algorithm 18 Find a data configuration for a table \mathbf{T} in cloud computing.

```

1: function BESTDATACONFIGURATION( $Q, \mathbf{W}, \mathbf{T}, \mathbf{S}, \mathbf{B}$ )
2:   // Find a Pareto data configuration set of table  $\mathbf{T}$  and Workload  $\mathbf{W}$  with weight
   sum model  $\mathbf{S}$  and Constraint  $\mathbf{B}$ 
3:    $\alpha \in \{0; 1\}$  //weight of similarity
4:    $\beta \in \{0; 1\}$  //clustering threshold
5:    $\theta \in \{0; 1\}$  //merging threshold
6:    $\lambda \in \{0; 1\}$  //data layout threshold
7:    $AUM \leftarrow AttributeUsageMatrix(W)$ 
8:    $F \leftarrow QueryFrequencies(W)$ 
9:    $I \leftarrow DataSpecific(T)$ 
10:   $P \leftarrow NSGA - G(\alpha, \beta, \theta, \lambda, AUM, I, F)$ 
11:  //Return best candidate in  $\mathcal{P}$  with weight sum model
12:  return  $BestInPareto(\mathcal{P}, \mathbf{S}, \mathbf{B})$ 
13: end function

```

Algorithm 19 Select the best configuration in \mathcal{G}

```

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

```

inal Pareto set by weighted sum model \mathbf{S} , as shown in Line 6. Finally, the best solution in Pareto set \mathcal{G} is selected by user weighted sum model S and constraint B .

7.5 Conclusion

In this chapter, we have presented the automatic generating data storage configuration to the cloud federation with MOOP. The automatic approach is proposed in HYTORMO [97], but the space of candidates is huge. We have extended HYTORMO to find an optimal data storage configuration for DICOM data storage. We have used NSGA-G to solve this problem to find an approximated optimal hybrid data storage configuration.

Our contributions, DREAM, NSGA-G and an application to optimal hybrid data stor-

age configuration for a cloud federation, have been presented in Chapter 5, 6, 7. In the next chapter, we implement our proposal approaches in cloud environment and compare them to other methods.

Key Points

- | |
|--|
| <ul style="list-style-type: none">• We have defined DICOM data management, and DICOM storage in a hybrid row-column system as a multi-objective problem.• We have used Non-dominated Sorting Genetic Algorithm based on Grid Partitioning to find the optimal data configuration for the hybrid data. |
|--|

Publication

- | |
|---|
| <ul style="list-style-type: none">• Trung-Dung Le, Verena Kantere, Laurent d'Orazio. Optimizing DICOM data management with NSGA-G. <i>International Workshop On Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP@EDBT)</i>, Lisbon, Portugal, 2019 |
|---|

PART III

Implementation of proposals and validations

PERFORMANCE VALIDATION

Contents

8.1 Introduction	108
8.2 A medical cloud federation	109
8.2.1 DICOM	109
8.2.2 Validation	109
8.3 DREAM	110
8.3.1 Implementation	110
8.3.2 Experiments	110
8.3.3 Results	112
8.4 NSGA-G	113
8.4.1 Validation on DTLZ test problems	113
8.4.2 Hybrid data storage configuration	127
8.5 Conclusion	134

8.1 Introduction

Chapters 5, 6, and 7 introduce two algorithms for the Multi-Objective Optimization Problem, and an approximate optimal solution for the management of DICOM data.

This chapter presents the validation of proposed algorithms and the implementation of the medical data system in a cloud federation. First, Section 8.2 introduces the implementation of the medical data, DICOM, in MIDAS. After that, experiments of DREAM with the TPC-H benchmark in IReS platform is presented and compared to other machine learning algorithms in Section 8.3. Our Multi-Objective Optimization Algorithm is validated with DTLZ test problems in Section 8.4. Two versions of NSGA-G are compared to other NSGAs in the multi-objective optimization problems. NSGA-G is

also applied to DICOM data to find an optimal solution in a MOOP. Finally, we conclude with section 8.5.

8.2 A medical cloud federation

As described in Chapter 3, the Medical Data Management System (MIDAS) is proposed to manage medical data. In particular, MIDAS extends the management of multiple engines platform, IReS, to organize DICOM in a cloud federation. This section presents more detail the implementation of DICOM data storage configurations in MIDAS.

8.2.1 DICOM

As introduced in Chapter 3, DICOM data set has been accessed by various OLAP, OLTP, and mixed workloads. Row stores data store all data associated with a row together. This strategy is suitable for OLTP workload, but wastes I/O costs for a query which requires few attributes of a table [66]. In contrast, column stores (e.g. MonetDB [19] and C-Store [125]) organize data by column. A column contains data for a single attribute of a tuple and stores sequentially on disk. The column stores allow reading only relevant attributes and efficiently aggregating over many rows, but only for a few attributes. Although the column stores are suitable for read-intensive (OLAP) workloads, their tuple reconstruction cost in OLTP workloads is higher than row stores. To improve the performance of storing and querying in OLAP, OLTP, and mixed workloads, DICOM data needs to be stored in a row-column store, called hybrid data storage. DICOM is used as an example of medical data in our system to validate the proposed approaches.

8.2.2 Validation

Modelling module requires low computation cost and accurate method to estimate the cost values. Machine learning methods can lead to the use of expired information. Hence, our proposed method, DREAM, is integrated into Modelling module to predict the cost values with low computation cost in a cloud environment. Moreover, the Multi-Objective Optimizer requires an efficient algorithm to find an approximation of

a Pareto-optimal solution. However, the space of candidate solutions in MOOP for DI-COM hybrid data is huge [97] and generating the Pareto-optimal front is often infeasible due to high complexity [160]. The vast space of data storage configuration candidates in a hybrid storage system leverages an alternative solution to find a Pareto-optimal one. Our method applies NSGA-G, an efficient MOO algorithm, to the Multi-Objective Optimizer to find an approximation of a Pareto-optimal solution. In conclusion, we propose to improve the accuracy of cost value prediction with low computation cost and to solve MOOP in both the querying and storing configurations with an efficient algorithm in a cloud environment.

Our validation¹ is implemented into IReS platform and a private cloud [127] with a cluster of three machines. The system uses Hadoop 2.7.3 [133] to organize the distributed data system. Beside the scenario of heterogeneous database system is implemented by three database engines. In particular, Hive 2.1.1 [129], PostgreSQL 9.5.14 [133], Spark 2.2.0 [135] are used in our system. The medical data management is run in Java OpenJDK Runtime Environment 1.8.0.

8.3 DREAM

8.3.1 Implementation

Our experiments are executed on a private cloud [127] with a cluster of three machines, as shown in Figure 8.1. 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 [133], Hive 2.1.1 [129], PostgreSQL 9.5.14 [133], Spark 2.2.0 [135] and Java OpenJDK Runtime Environment 1.8.0. IReS platform is used to manage data in multiple database engines and deploy the algorithms. We choose this implementation because it shows the simple architecture of MIDAS with 3 database engines and a single cluster.

8.3.2 Experiments

TPC-H benchmark [137] with two datasets of 100MB and 1GB is used to have experiments with DREAM. Experiments with TPC-H benchmark are executed in a multi-

1. <https://gitlab.inria.fr/trle/IReS-Platform>

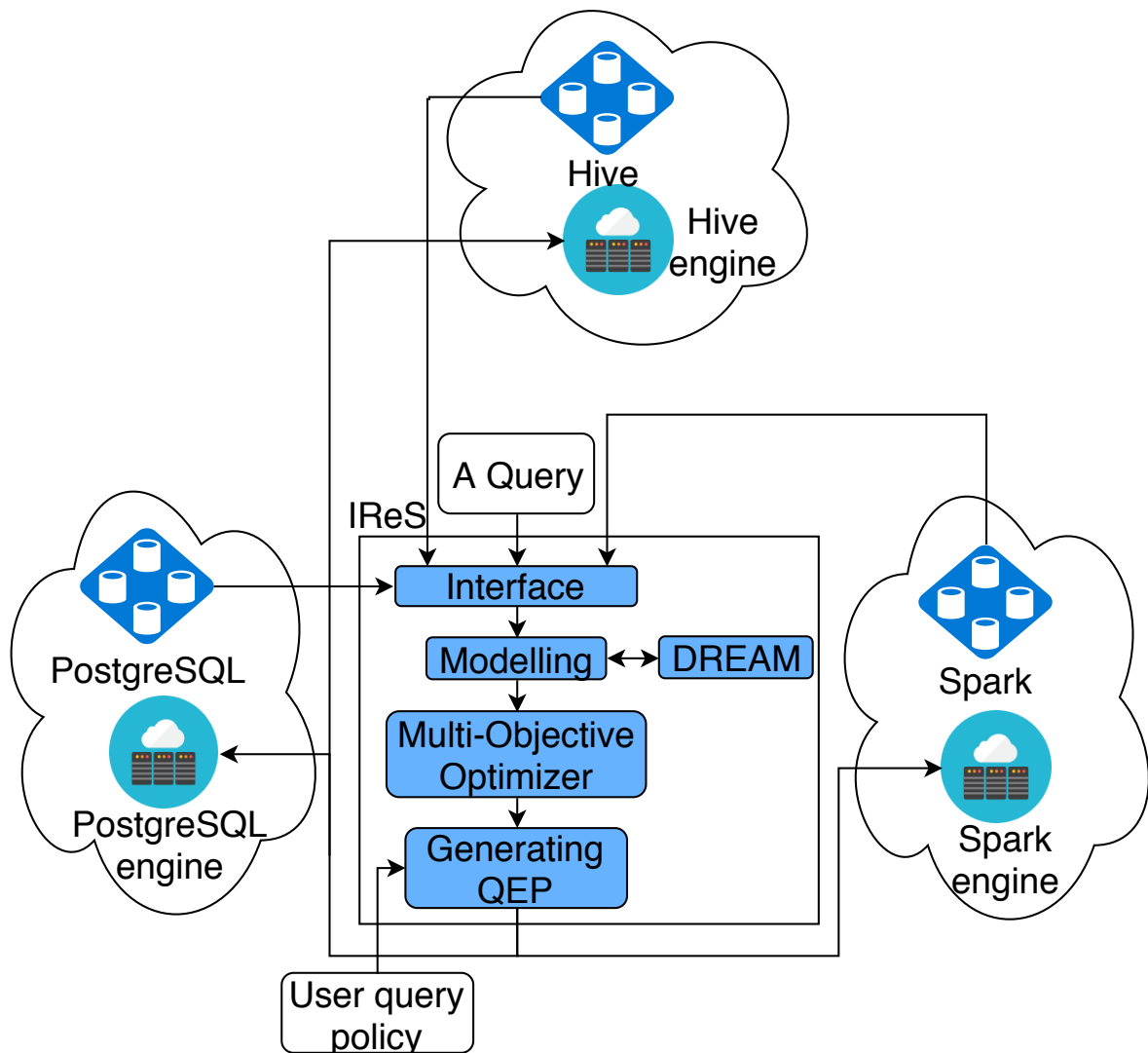


Figure 8.1 – Implementation of MIDAS, DREAM and Multi-Objective Optimizer.

engine environment consisting of Hive[129] and PostgreSQL[133] deployed on a private cloud [127].

In the TPC-H benchmark, the queries related to two tables are 12, 13, 14 and 17. We choose these queries because it shows the simple circumstance of joining tables with different database engines in cloud environment. These queries with two tables in two different databases, such as Hive and PostgreSQL, are studied. In our experiments, a query is selected randomly in these four queries. Besides, the database engines which store tables related to queries are chosen randomly. For example, a

Table 8.1 – Comparison of mean relative error with 100MiB TPC-H dataset.

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

query is selected randomly in Query 12, 13, 14, 17, and then the result is Query 12. Two tables related to Query 12 are orders, and lineitem. There are two circumstances: (1) The database engine managing orders table is Hive, and PostgreSQL stores lineitem table; (2) The lineitem table is stored in Hive, and PostgreSQL stores orders table. The circumstance is selected randomly, too.

8.3.3 Results

To estimate the quality of DREAM in comparison with other algorithms, Mean Relative Error (MRE) [4] is used and described as below:

$$\frac{1}{M} \sum_{i=1}^M \frac{|\hat{c}_i - c_i|}{c_i}, \quad (8.1)$$

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 the estimating process. DREAM is compared to the Best Machine Learning model (BML) in IReS platform with many observation window (N , $2N$, $3N$ and entire data samples). The smallest size of a window, $N = L + 2$ [121], where L is the number of variables, is the minimum data set DREAM requires.

In Table 8.1 and 8.2, IReS uses various machine learning algorithms to build cost models and choose the best model which has the smallest Mean Relative Error (BML). *BML* means that the entire historic data is used to train the cost models. *BML_N* means the size of the observation window is N . The lower value of MRE in two tables

Table 8.2 – Comparison of mean relative error with 1GiB TPC-H dataset.

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

indicates the more accurately estimating cost model. As shown in two tables, MRE of DREAM is the smallest values between various observation windows. For example, the estimation execution times of query 12 in both tables show that DREAM estimates more accurate than other algorithms. The best machine learning algorithm is the algorithm IReS selected which have the smallest MRE. There are various versions of BML according to the size of observation window, such as N , $2N$, $3N$, and entire data samples. In our experiments, the size of data samples, which DREAM uses, are very small, around N . However, the MRE of DREAM is the smallest value among them.

8.4 NSGA-G

8.4.1 Validation on DTLZ test problems

Various earlier studies on Multiple Objective Evolutionary Algorithms (MOEAs) introduce test problems which are either simple or not scalable. DTLZ test problems [41] 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.

Environment

For fair comparison and evaluation, the same parameters are used, such as Simulated binary crossover (30), Polynomial mutation (20), max evaluations (10000) and populations (100) for eMOEA[31], NSGA-II, MOEA/D[158], NSGA-III and NSGA-G²,

2. <https://gitlab.inria.fr/trle/moea>

during their 50 independent runs to solve two types of problems in DTLZ test problems [41] with m objectives, $m \in [5, 10]$ in Multiobjective Evolutionary Algorithms (MOEA) framework [130] in Open JDK Java 1.8. These algorithms use the same population size $N = 100$ and the maximum evaluation $M = 10000$. All experiments are run on a machine with following parameters: Intel(R) core(TM) i7-6600U CPU @ 2.60GHz \times 4, 16GB RAM.

NSGA-G with Min point

To estimate the qualities of algorithms, the Generational Distance (GD) [148], Inverted Generational Distance (IGD) [30] and the Maximum Pareto Front Error (MPFE) [150] are applied.

GD measures how far the evolved solution set is from the true Pareto front [155], as shown in following:

$$GD = \frac{\sqrt{\sum_{i=1}^n d_i^2}}{n}, \quad (8.2)$$

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 [15], which can measure both convergence and diversity in a sense. IGD is shown in the following equation [155]:

$$IGD = \frac{\sum_{v \in PF_{true}} d(v, X)}{|PF_{true}|}, \quad (8.3)$$

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} 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 [155]. This metric is shown in the following equation:

$$MPFE = \max_i d_i. \quad (8.4)$$

Table 8.3 – Generational Distance

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	3.675e-02	4.949e+01	1.129e-01	2.494e+00	2.721e-03
DTLZ3	5	1.030e-01	4.418e+00	1.951e-01	7.214e-01	6.342e-03
DTLZ1	6	1.600e-01	9.637e+01	3.138e-01	1.049e+00	3.850e-02
DTLZ3	6	1.306e+01	1.289e+02	5.265e+00	9.577e+00	9.921e-01
DTLZ1	7	1.390e-01	5.283e+01	1.515e-01	4.515e-01	1.542e-02
DTLZ3	7	3.793e-01	3.714e+00	2.251e-02	1.600e-01	2.379e-03
DTLZ1	8	6.817e-01	1.175e+02	2.608e-01	1.949e+00	8.223e-02
DTLZ3	8	1.419e+01	1.667e+02	5.320e+00	1.351e+01	9.146e-01
DTLZ1	9	4.451e-01	4.808e+01	1.101e-01	1.917e+00	1.040e-02
DTLZ3	9	6.843e-02	1.620e+00	5.237e-03	1.280e-01	1.325e-03
DTLZ1	10	3.431e-01	4.340e+01	1.432e-01	2.115e+00	0.000e+00
DTLZ3	10	8.458e-02	1.593e+00	6.763e-03	1.627e-01	1.815e-03

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.

By dividing the space of solutions into multiple partitions and selecting groups randomly, the proposed algorithm has both advantages of diversity and convergence, comparing to other NSGAs. The advantages of NSGA-G are not only on GD, IGD, as shown in Tables 8.3 and 8.5, but also in MPFE experiment, as presented in Table 8.7. As shown Equation 8.2, 8.3, the lower values of GD and IGD indicate the better performance of algorithms. The convergence and diversity of NSGA-G are often the most or second quality in the tests. Equation 8.4 shows that the small of the most significant distance between the individuals in Pareto front and the solutions in the approximation front is prefer than the big ones. The MPFE values in Table 8.7 is also shown that the worst significant distance between solutions in the approximate front and the individuals in true Pareto front of NSGA-G is smaller than others one among algorithms.

By comparing solutions in a group, instead of all the space, the proposed algorithm outperforms to other NSGAs in terms of computation time, as shown in Tables 8.4, 8.6 and 8.8. It can be seen that the computation time of NSGA-G is better than the others among algorithms in most cases.

Table 8.4 – Average compute time (seconds) in Generational Distance experiment

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	5.904e+01	1.063e+02	2.264e+02	4.786e+02	1.261e+02
DTLZ3	5	1.005e+02	1.111e+02	2.358e+02	5.040e+02	1.233e+02
DTLZ1	6	9.024e+01	1.089e+02	2.320e+02	3.509e+02	1.083e+02
DTLZ3	6	1.602e+02	1.243e+02	2.520e+02	3.653e+02	1.209e+02
DTLZ1	7	1.038e+02	1.200e+02	2.839e+02	3.986e+02	1.244e+02
DTLZ3	7	2.946e+02	1.381e+02	2.820e+02	3.565e+02	1.342e+02
DTLZ1	8	1.463e+02	1.313e+02	2.896e+02	4.926e+02	1.249e+02
DTLZ3	8	5.575e+02	1.541e+02	3.458e+02	5.633e+02	1.399e+02
DTLZ1	9	1.573e+02	1.428e+02	3.242e+02	6.823e+02	1.496e+02
DTLZ3	9	8.147e+02	1.988e+02	3.721e+02	8.136e+02	1.640e+02
DTLZ1	10	1.436e+02	1.611e+02	3.745e+02	9.589e+02	1.370e+02
DTLZ3	10	9.151e+02	1.801e+02	3.907e+02	9.805e+02	1.577e+02

Table 8.5 – Inverted Generational Distance

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	4.070e-01	8.247e+01	3.434e-01	2.796e+00	3.314e-01
DTLZ3	5	1.656e-01	6.364e+00	3.335e-01	1.383e+00	1.922e-01
DTLZ1	6	7.981e-01	1.786e+02	9.150e-01	3.040e+00	7.034e-01
DTLZ3	6	4.429e+01	4.526e+02	1.164e+01	3.103e+01	8.100e+00
DTLZ1	7	4.188e-01	2.203e+01	3.280e-01	5.024e-01	3.715e-01
DTLZ3	7	9.630e-01	9.286e+00	1.929e-01	3.901e-01	1.667e-01
DTLZ1	8	1.417e+00	2.691e+02	1.023e+00	4.195e+00	9.540e-01
DTLZ3	8	1.023e+02	6.471e+02	1.167e+01	4.194e+01	7.513e+00
DTLZ1	9	4.432e-01	2.396e+01	3.019e-01	6.685e-01	3.147e-01
DTLZ3	9	3.737e-01	3.368e+00	1.381e-01	2.516e-01	1.331e-01
DTLZ1	10	5.912e-01	1.723e+01	3.737e-01	8.963e-01	3.613e-01
DTLZ3	10	6.287e-01	6.049e+00	1.296e-01	5.049e-01	1.521e-01

Table 8.6 – Average compute time (seconds) in Inverted Generational Distance experiment

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	6.780e+01	9.430e+01	2.292e+02	4.564e+02	9.646e+01
DTLZ3	5	9.976e+01	1.156e+02	2.564e+02	5.036e+02	1.166e+02
DTLZ1	6	7.696e+01	1.078e+02	2.451e+02	3.471e+02	1.178e+02
DTLZ3	6	1.549e+02	1.300e+02	2.527e+02	3.714e+02	1.986e+02
DTLZ1	7	1.021e+02	1.286e+02	2.732e+02	3.271e+02	1.297e+02
DTLZ3	7	3.522e+02	1.942e+02	3.794e+02	3.582e+02	1.523e+02
DTLZ1	8	1.170e+02	1.292e+02	3.222e+02	4.677e+02	1.212e+02
DTLZ3	8	5.333e+02	1.526e+02	3.140e+02	5.190e+02	1.431e+02
DTLZ1	9	1.435e+02	1.812e+02	3.120e+02	7.548e+02	1.544e+02
DTLZ3	9	7.445e+02	2.171e+02	3.533e+02	7.884e+02	1.485e+02
DTLZ1	10	2.104e+02	1.786e+02	3.942e+02	1.532e+03	2.182e+02
DTLZ3	10	1.195e+03	2.526e+02	5.766e+02	1.302e+03	2.131e+02

Table 8.7 – Maximum Pareto Front Error

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	7.363e-01	8.969e+02	2.556e+00	2.260e+02	1.024e-01
DTLZ3	5	9.455e+00	1.015e+02	3.692e+00	4.002e+01	1.957e-01
DTLZ1	6	4.699e+00	1.584e+03	8.950e+00	7.488e+01	3.375e-01
DTLZ3	6	5.112e+02	1.862e+03	9.387e+01	4.340e+02	1.244e+01
DTLZ1	7	9.524e+00	1.012e+03	3.074e+00	1.802e+01	1.695e-01
DTLZ3	7	1.458e+01	3.163e+01	2.035e-01	3.116e+00	2.708e-02
DTLZ1	8	3.186e+01	2.041e+03	5.685e+00	2.127e+02	5.532e-01
DTLZ3	8	1.170e+03	2.247e+03	9.867e+01	5.268e+02	1.145e+01
DTLZ1	9	1.111e+01	1.036e+03	2.075e+00	1.496e+02	3.106e-01
DTLZ3	9	1.320e+01	4.065e+01	1.354e-01	8.366e+00	3.195e-02
DTLZ1	10	2.641e+01	1.026e+03	2.793e+00	2.293e+02	0.000e+00
DTLZ3	10	1.492e+01	4.185e+01	1.368e-01	1.079e+01	2.744e-02

Table 8.8 – Average compute time (seconds) in Maximum Pareto Front Error experiment

	m	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
DTLZ1	5	7.454e+01	1.214e+02	2.742e+02	5.796e+02	1.221e+02
DTLZ3	5	1.231e+02	1.437e+02	3.118e+02	6.035e+02	1.286e+02
DTLZ1	6	1.040e+02	1.318e+02	2.848e+02	4.258e+02	1.276e+02
DTLZ3	6	2.166e+02	1.673e+02	3.462e+02	5.014e+02	1.575e+02
DTLZ1	7	1.276e+02	1.638e+02	3.230e+02	4.314e+02	1.424e+02
DTLZ3	7	4.594e+02	1.959e+02	4.188e+02	5.557e+02	1.774e+02
DTLZ1	8	1.637e+02	1.609e+02	3.832e+02	5.952e+02	1.466e+02
DTLZ3	8	5.940e+02	1.963e+02	3.640e+02	6.025e+02	1.453e+02
DTLZ1	9	1.369e+02	1.474e+02	3.148e+02	7.728e+02	1.559e+02
DTLZ3	9	6.596e+02	1.982e+02	3.984e+02	8.069e+02	1.516e+02
DTLZ1	10	1.546e+02	1.540e+02	3.555e+02	9.331e+02	1.400e+02
DTLZ3	10	8.219e+02	1.841e+02	3.601e+02	9.677e+02	1.619e+02

NSGA-G with Random metric

Similarly as NSGA-G with Min point, we use the Generational Distance (GD) [148], Inverted Generational Distance (IGD) [30] and the Maximum Pareto Front Error (MPFE) [150] to compare the quality of NSGA-G with Random metric (NSGA-GR) to other NSGAs, including NSGA-G with Min point.

At this section, besides of using DTLZ problems, we use WFG [69] test problem. Advantages of two versions of NSGA-G are present in tables 8.9, 8.10, 8.11, 8.12, 8.13, and 8.14. Similar to the previous section, the measuring metrics, such as GD, IDG, MPFE, are used to estimate the qualities of the different algorithms. These experiment compare NSGA-G with Random metric to other algorithms, including NSGA-G with Min point.

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, 8.12, 8.14.

Second, NSGA-Gs are also better than other NSGAs in terms of quality in GD and MPFE experiments, as shown in Table 8.9, 8.13. Except for the IDG experiment, as shown in Table 8.11 the quality of NSGA-G with Random metric is not as good as other ones. However, the fastest algorithm among NSGAs is often NSGA-G with Random metric. It can be accepted for the trade-off between quality and computation time.

Table 8.9 – 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

Table 8.10 – 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 8.11 – 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 8.12 – 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 8.13 – 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 8.14 – Average compute time in Maximum Pareto Front Error experiment

	m	eMOEA	NSGAR	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

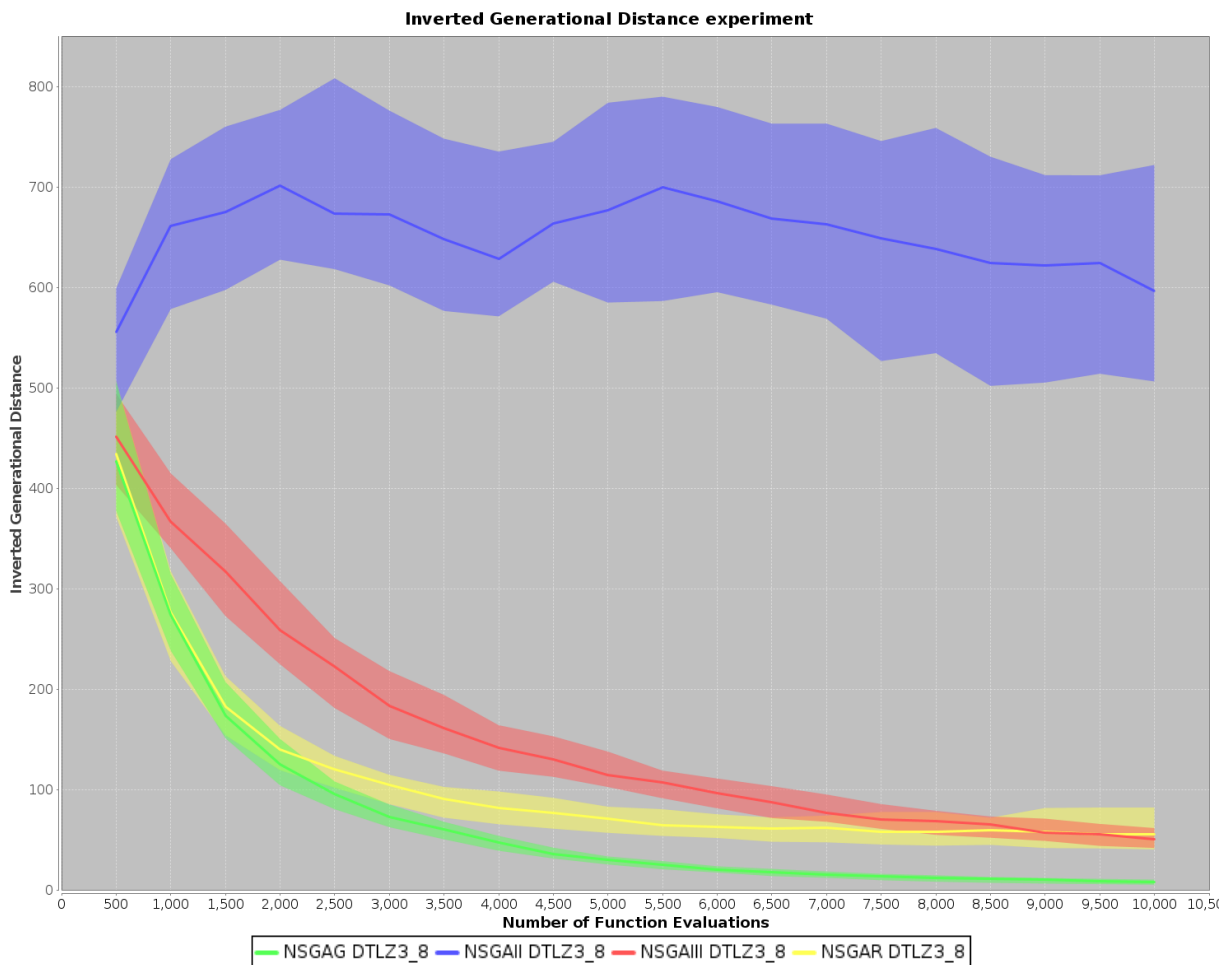


Figure 8.2 – Inverted Generational Distance of 4 algorithms with DTLZ3_8.

Experiment of changing the evaluation

In two 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, call 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 NSGAG and NSGAR, respectively. The results in Figure 8.2 and 8.3 show the advantages of two versions of NSGA-G. Both their convergence and diversity are better than NSGA-II and NSGA-III. They are also faster than others.

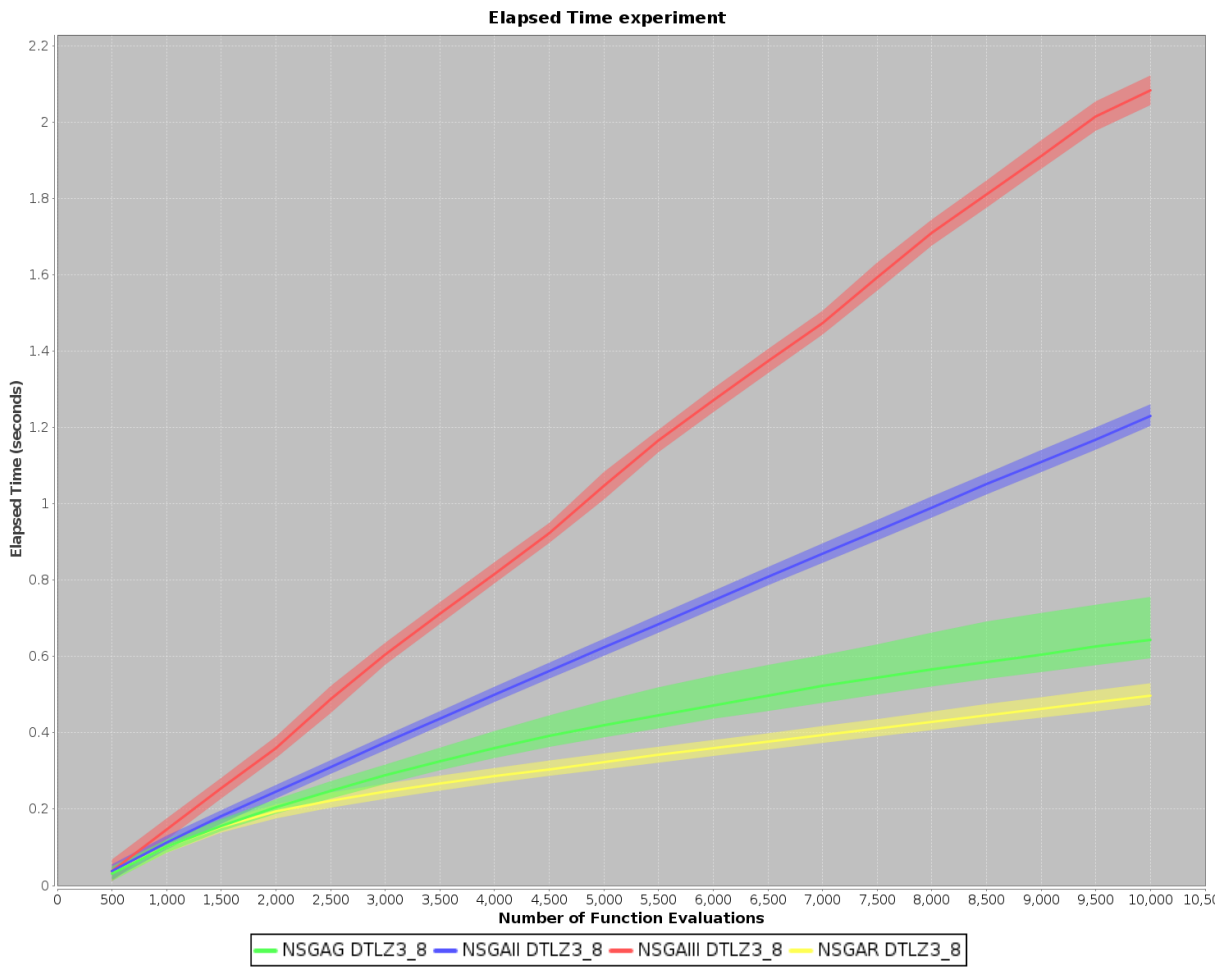


Figure 8.3 – Execution time of 4 algorithms with DTLZ3_8.

Table 8.15 – Example of real DICOM data set.

Datasets	DICOM files	AttributesTuples	Metadata	Total size
CTColonography	98,737	86	7.76 GB	48.6 GB
Dclunie	541	86	86.0 MB	45.7 GB
Idoimaging	1,111	86	53.9 MB	369 MB
LungCancer	174,316	86	1.17 GB	76.0 GB
MIDAS	2,454	86	63.4 MB	620 MB
CIAD	3,763,894	86	61.5 GB	1.61 TB

Table 8.16 – Example of extracted DICOM data set.

Table	Number of Tuples	Size
Patient	120,306	20.788 MB
Study	120,306	19.183 MB
GeneralInfoTable	16,226,762	4,845,042 MB
SequenceAttributes	4,149,395	389.433 MB

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.

8.4.2 Hybrid data storage configuration

In this section, the proposed algorithm, NSGA-G, is applied to DICOM dataset to look for a Pareto set of data storage configurations. The dataset containing the DICOM files in the white paper by Oracle [105] is created by six different digital imaging modalities. Its total size is about 2 terabytes, including 2.4 million images of 20,080 studies. In particular, DICOM text files are used in [97], as shown in Table 8.16. They are extracted from real DICOM dataset, as shown in Table 8.15. The extracted DICOM dataset [97] comprises four tables: GeneralInfoTable, SequenceAttributes, Patient, Study.

Patient table

Patient table extracted from DICOM data has 120,306 tuples and 20.788 MB. It is often processed by a workload W_P , as shown in Table 8.17. The Attribute Usage Matrix of Patient table is shown in Table 8.18. The null ratios of the attributes of the entity Patient table are:

- PatientName: 0.0%,
- PatientID: 0.0%,
- PatientBirthDate: 83.55%,
- PatientSex: 1.48%,
- EthnicGroup: 100%,
- IssuerOfPatientID: 100%,
- PatientBirthTime: 96.32%,
- PatientInsurancePlanCodeSequence: 100%,
- PatientPrimaryLanguageCodeSequence: 100%,
- PatientPrimaryLanguageModifierCodeSequence: 100%,
- OtherPatientIDs: 100%,
- OtherPatientNames: 100%,
- PatientBirthNames: 100%,
- PatientTelephoneNumbers: 100%,
- SmokingStatus: 97.48%,
- PregnancyStatus: 90.01%,
- LastMenstrualDate: 97.72%,
- PatientReligiousPreference: 100%,
- PatientComments: 99.64%,
- PatientAddress: 100%,
- PatientMotherBirthName: 100%,
- InsurancePlanIdentification: 100%.

Study table

Study table extracted from DICOM data has 120,306 tuples and 19.183 MB. Workload W_S accessing Study table is shown in Table 8.19. The Attribute Usage Matrix of Study table is shown in Table 8.20. The null ratios of the attributes of the entity Study table are:

- StudyInstanceUID: 0.0%,
- StudyDate: 0.07%,
- StudyTime: 0.07%,
- ReferringPhysicianName: 16.44%,
- StudyID: 15.65%,
- AccessionNumber: 93.93%,

Table 8.17 – Frequency of Queries in Workload W_P .

Queries	Detail	Freq
Q_{p1}	SELECT UID, PatientName, PatientID, PatientBirthDate, PatientTelephoneNumbers, PatientSex, PatientBirthName, SmokingStatus, PatientComments, PatientMotherBirthName FROM Patient WHERE PatientID = 'P30013'	300
Q_{p2}	SELECT UID, PatientName, PatientID, PatientBirthDate, PatientSex, EthnicGroup, IssuerOfPatientID, OtherPatientNames, PatientMotherBirthName, InsurancePlanIdentification FROM Patient	100
Q_{p3}	SELECT UID, PatientID, PatientName, PatientBirthDate, PatientSex, EthnicGroup, SmokingStatus FROM Patient WHERE PatientSex = 'M' AND SmokingStatus = 'NO'	100
Q_{p4}	SELECT UID, PatientName, PatientID, PatientBirthDate, EthnicGroup, PatientPrimaryLanguageModifierCodeSequence, OtherPatientIDs, PatientAddress FROM Patient	100
Q_{p5}	SELECT UID, PatientName, PatientID, PatientBirthDate, PatientBirthTime, PatientInsurancePlanCodeSequence, PregnancyStatus, LastMenstrualDate, PatientReligiousPreference FROM Patient	100
Q_{p6}	SELECT UID, PatientName, PatientID, PatientBirthDate, EthnicGroup, PregnancyStatus, LastMenstrualDate FROM Patient	100

Table 8.18 – Attribute Usage Matrix of Patient table.

Attributes	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	Q ₆
PatientName	1	1	1	1	1	1
PatientID	1	1	1	1	1	1
PatientBirthDate	1	1	1	1	1	1
PatientSex	1	1	1	0	1	0
EthnicGroup	0	1	1	1	0	1
IssuerOfPatientID	0	1	0	0	0	0
PatientBirthTime	0	0	0	0	1	0
PatientInsurancePlanCodeSequence	0	0	0	0	1	0
PatientPrimaryLanguageCodeSequence	0	0	1	0	0	0
PatientPrimaryLanguageModifierCodeSequence	0	0	0	1	0	0
OtherPatientIDs	0	0	0	1	0	0
OtherPatientNames	0	1	0	0	0	0
PatientBirthNames	1	0	0	0	0	0
PatientTelephoneNumbers	1	0	0	0	0	0
SmokingStatus	1	0	1	0	0	0
PregnancyStatus	0	0	0	0	1	1
LastMenstrualDate	0	0	0	0	1	1
PatientReligiousPreference	0	0	0	0	1	0
PatientComments	1	0	0	0	0	0
PatientAddress	0	0	0	1	0	0
PatientMotherBirthName	1	1	0	0	0	0
InsurancePlanIdentification	0	1	0	0	0	0

- StudyDescription: 0.48%,
- PatientAge: 11.23%,
- PatientWeight: 14.18%,
- PatientSize: 90.34%,
- Occupation: 99.63%,
- AdditionalPatientHistory: 71.64%,
- MedicalRecordLocator: 100%,
- MedicalAlerts: 100%.

GeneralInfoTable

GeneralInfoTable table is extracted from DICOM data. It is often processed by a workload **W**, as shown in Table 8.21. The Attribute Usage Matrix of GeneralInfoTable

Table 8.19 – Frequency of Queries in Workload W_s .

Queries	Detail	Freq
Q_{s1}	SELECT StudyInstanceUID, StudyDate, StudyTime, ReferringPhysicianName, StudyID, AccessionNumber, MedicalAlerts FROM Study WHERE StudyDate >= '20000101' AND StudyDate <= '20150101'	300
Q_{s2}	SELECT StudyInstanceUID, StudyDate, StudyTime, ReferringPhysicianName, StudyID, MedicalRecordLocator FROM Study WHERE StudyID = '20050920'	100
Q_{s3}	SELECT PatientAge, PatientWeight, PatientSize FROM Study WHERE PatientAge >= 90 Q4,4s	100
Q_{s4}	SELECT UID, StudyInstanceUID, StudyDate, StudyTime, ReferringPhysicianName, StudyID, AccessionNumber, PatientWeight, AdditionalPatientHistory FROM Study	100
Q_{s5}	SELECT StudyInstanceUID, StudyDate, StudyTime, StudyID, PatientSize, Occupation FROM Study	100
Q_{s6}	SELECT StudyInstanceUID, StudyDate, StudyTime, ReferringPhysicianName, StudyID, StudyDescription, PatientAge FROM Study WHERE StudyDate >= '20000101' AND StudyDate <= '20150101'	100

Table 8.20 – Attribute Usage Matrix of Study table.

Attributes	Q_{s1}	Q_{s2}	Q_{s3}	Q_{s4}	Q_{s5}	Q_{s6}
StudyInstanceUID	1	1	0	1	1	1
StudyDate	1	1	0	1	1	1
StudyTime	1	1	0	1	1	1
ReferringPhysicianName	1	1	0	1	0	1
StudyID	1	1	0	1	1	1
AccessionNumber	1	0	0	1	0	0
StudyDescription	0	0	0	0	0	1
PatientAge	0	0	1	0	0	1
PatientWeight	0	0	1	1	0	0
PatientSize	0	0	0	0	1	0
Occupation	0	0	0	0	1	0
AdditionalPatientHistory	0	0	0	1	0	0
MedicalRecordLocator	0	1	0	0	0	0
MedicalAlerts	1	0	0	0	0	0

Table 8.21 – Frequency of Queries in Workload W_{Ge} .

Queries	Detail	Freq
Q_{Ge_1}	SELECT UID, GeneralTags, GeneralVRs, GeneralNames, GeneralValues FROM GeneralInfoTable	100
Q_{Ge_2}	SELECT GeneralTags, count(GeneralValues) FROM GeneralInfoTable GROUP BY GeneralTags	100
Q_{Ge_3}	SELECT UID, GeneralNames FROM GeneralInfoTable WHERE GeneralNames = 'Modality'	100
Q_{Ge_4}	SELECT UID, GeneralVRs FROM GeneralInfoTable WHERE GeneralVRs = 'DA'	100

Table 8.22 – Attribute Usage Matrix of GeneralInfoTable.

Queries	GeneralTags (a_1)	GeneralVRs (a_2)	GeneralNames (a_3)	GeneralValues (a_4)
Q_{Ge_1}	1	1	1	1
Q_{Ge_2}	1	0	0	1
Q_{Ge_3}	0	0	1	0
Q_{Ge_4}	0	1	0	0

table is shown in Table 8.22. GeneralInfoTable has four attributes with the null ratios of the attributes, given by:

- GeneralTags: 0.0%,
- GeneralVRs: 0.0%,
- GeneralNames: 0.0%,
- GeneralValues: 13.97%.

SequenceAttributes

SequenceAttributes table [97] is extracted from DICOM data. It is often processed by a workload W_{Seq} , as shown in Table 8.23. The Attribute Usage Matrix related to SequenceAttributes is shown in Table 8.24. SequenceAttributes has four attributes with the null ratios of the attributes as follows:

- SequenceTags: 0.0%,
- SequenceVRs: 0.0%,
- SequenceNames: 0.0%,
- SequenceValues: 0.34%.

Table 8.23 – Frequency of Queries in Workload W_{Seq} .

Queries	Detail	Freq
Q_{Seq1}	SELECT UID, SequenceTags, SequenceVRs, SequenceNames, SequenceValues FROM SequenceAttributes WHERE SequenceNames LIKE '%X-Ray%'	100
Q_{Seq2}	SELECT SequenceTags, SequenceVRs, SequenceNames FROM SequenceAttributes WHERE SequenceVRs = 'CS'	100

Table 8.24 – Attribute Usage Matrix of SequenceAttributes.

Queries	SequenceTags (a_1)	SequenceVRs (a_2)	SequenceNames (a_3)	SequenceValues (a_4)
Q_{Seq1}	1	1	1	1
Q_{Seq2}	1	1	1	0

Results

The number of attributes in GeneralInfoTable and SequenceAttributes is four and the null ratios of them often equal to 0.0%. Hence, the number of data storage configuration candidates is not too big. The experiments give the same results in GD and IDG quality tests with these two tables.

On the other hand, the information of Patient and Study tables are more complicated than the others in DICOM. NSGA-G and other NSGAs are experimented with Patient and Study tables in GD and IGD quality tests. The version of NSGA-G we use in this section is NSGA-G with Min point. These algorithms use the same population of size $N = 100$ and the maximum evaluation $M = 100$, while the default values in MOEA framework are used, such as Simulated binary crossover (30) and Polynomial mutation (20). Tables 8.25 and 8.26 show the qualities of diversity and convergence of five algorithms. As mentioned in previous Section 8.4.1, the lower value of experiments represents a better quality of an algorithm. In this experiment, the best algorithm is NSGA-III and the second one is NSGA-G. These results can be explained by the fact that the DICOM data storage configuration problem is less complicated than DTLZ problems. Moreover, Table 8.28 show the advantage of NSGA-G among five NSGAs in execution times.

In conclusion, despite the best quality algorithm in the case of DICOM hybrid store, the computation time of NSGA-III is too long. In contrast, in spite of the second good algorithm, the execution time of NSGA-G is shorter than the others.

Table 8.25 – Generational Distance.

	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
Patient	1.997e-02	2.156e-02	2.289e-02	1.604e-02	1.853e-02
Study	6.495e-02	6.166e-02	6.210e-02	5.559e-02	7.476e-02

Table 8.26 – Inverted Generational Distance.

	eMOEA	NSGA-II	MOEA/D	NSGA-III	NSGA-G
Patient	9.816e-02	1.002e-01	9.922e-02	8.552e-02	9.796e-02
Study	4.636e-02	4.445e-02	6.509e-02	4.249e-02	4.374e-02

Table 8.27 – Maximum Pareto Front Error

	eMOEA	NSGAI	MOEAD	NSGAI	NSGAG
patientall	2.492e-01	2.492e-01	2.492e-01	2.492e-01	2.492e-01
studyall	4.888e-01	6.027e-01	6.931e-01	6.563e-01	6.956e-01

Table 8.28 – The execution time of NSGAs with DICOM.

Table	eMOEA(s)	NSGA-II(s)	MOEA/D(s)	NSGA-III(s)	NSGA-G(s)
Patient	17.804	17.822	17.810	17.907	17.740
Study	7.659	7.720	7.775	7.718	7.706

8.5 Conclusion

This chapter experiments the medical data management, MIDAS, in a cloud federation. In particular, DICOM dataset is organized by the management of multiple engines platform, IReS, in the cloud environment. Dynamic REgression Algorithm (DREAM) and Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) are introduced as a part of Medical Data Management System and on top of IReS.

Furthermore, this chapter presents experiments of our solution to optimize storage and query processing of DICOM files in a hybrid (row-column) store. The experiments validate DREAM with IReS platform, and compare the quality of DREAM to other machine learning algorithms in IReS with TPC-H benchmark. NSGA-Gs are experimented on DTLZ, WFG test problems. Besides, NSGA-G is used to find an approximation of Pareto-optimal with a good trade-off between diversity and performance. Preliminary experiments on DICOM files in a hybrid store prove that NSGA-G also provides the best processing time with interesting results in both diversity and convergence.

Key Points

- We execute the experiments to validate DREAM with IReS platform, and compare the quality of DREAM to other machine learning algorithms in IReS with TPC-H benchmark.
- We execute the experiments to validate NSGA-G with MOEA framework, and compare the quality of NSGA-Gs to other NSGAs with DTLZ, and WFG problems.
- We execute the experiments to validate NSGA-G to find the approximate optimal solution in the hybrid data storage configuration with DICOM dataset.

CONCLUSION AND FUTURE WORK

Contents

9.1 Introduction	137
9.2 Summary and Conclusion	137
9.2.1 Existing solutions	138
9.2.2 Estimating in Multi-Objective Optimization Problem	139
9.2.3 Multi-Objective Evolutionary Algorithm	140
9.2.4 Optimizing medical data storage configuration	140
9.3 Future Works	141
9.3.1 Estimation	141
9.3.2 Searching and optimization	141
9.3.3 Hybrid data storage configuration	142

9.1 Introduction

This chapter concludes our work on data management in cloud federations and its application in the medical domain. Section 9.2 summarizes our contributions. Large scale data management is a vast domain that we have very partially addressed. Section 9.3 describes future research directions in estimation, searching and optimization for MOOPs, their applications.

9.2 Summary and Conclusion

Medical data management in cloud federations raises Multi-Objective Optimization Problems (MOOPs) for query processing and data storage, according to users preferences, such as response time, monetary cost, qualities, etc. These problems are

complex to address because of the variability of the environment (due to virtualization, large-scale communications, etc.). Hence, the cost values for MOOPs need to be estimated accurately by using efficient data sample. Besides, MOOP in a cloud federation needs to be solved by an efficient search and optimization method. In particular, data storage configuration for DICOM files on a hybrid row-column system need to find an optimal solution.

Five main contributions emerged from this dissertation. First, we synthesize a comprehensive evaluation of the existing solutions related to query processing and data storage in cloud federations. The state of the art also presented the current research on medical data managing methods in row, column, row-column storages. The searching and optimization methods are described with their strengths and weaknesses. Second, we proposed a dynamic multiple linear regression method to estimate accurately cost value in the cloud environment. Third, an efficient Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) is proposed to search and optimize the process to solve Multi-Objective Optimization Problem in a cloud federation. We then use NSGA-G to find an approximate optimal solution of a data storage configuration for DICOM files on a hybrid row-column system. Finally, we performed validations to demonstrate the advantages of our proposed approaches.

9.2.1 Existing solutions

Medical data in a cloud federation raises challenges in data management in terms of Multi-Objective Optimization in hybrid data storage configuration, query processing, the management of multiple engines platform.

First, the thesis shows recent works on various aspects in cloud environment, i.e., resource and data management. With regard to cloud environment, the existing systems focus on optimizing single metrics, such as execution time, data transferring. They do not consider the MOOP in the optimization process. IReS is to the best of our knowledge the only platform that addresses this kind of problem, but the algorithm they use to built the cost model and optimize MOOP has limited performance. Therefore, we reuse IReS, which is an open source, in a multi-engine environment, and evaluate the proposal approaches to solve MOOP in a cloud federation more efficiently.

Second, based on the specific characteristics of medical data, we discussed opportunities and challenges in terms of data storage and possible layout configurations.

DICOM standard and medical data management using this format is also presented. We described recent works on hybrid data storage configuration approaches. HY-TORMO [97] addresses an automatically generate hybrid data storage configuration in terms of query processing, data storage, the high volume and sparsity of DICOM data. However, they do not give any method to find the optimal hybrid data storage configurations [97, 101] in MOOP. Hence, in the context of MOOP in HYTORMO and the huge space of candidates, the hybrid data configuration needs find an approximate optimal solution with a low computational complexity Multi-Objective algorithm.

Third, the goal of Multi-Objective Optimization (MOO) is to find the set of Pareto-optimal solutions, i.e. the query plans realizing optimal cost trade-offs for a given query, the hybrid data storage configuration optimal cost trade-offs for a given workload and data set. The MOO background is presented. Pareto set aspect and the widely estimation technique, Multiple Linear Regression, used in science and engineering are showed in the thesis. It is introduced to build a cost model for estimating multiple cost value in MOOP. Related to searching and optimizing approaches, we focus on the process of a good technique in MOO, such as EMO class. In particular, NSGAs show their advantage in EMO algorithms. However, the characteristics of NSGAs should be improved to have better performance, such as diversity, convergence. Therefore, the MOOP in cloud environment need to have efficient algorithms to build cost models, and optimize query processing and data storage.

9.2.2 Estimating in Multi-Objective Optimization Problem

We proposed Dynamic REgression Algorithm (DREAM) to improve the accuracy of cost values in a cloud federation. The proposed algorithm reduces the size of historical data in the training process based on the coefficient of determination. Following this approach the execution time for training and testing process is decreased. This characteristic is suitable to the variability of cloud environments, such as users demands of resources.

We validated DREAM with IReS platform, and compare the quality of DREAM to other machine learning algorithms with TPC-H benchmark. The experiments show that DREAM estimations are more accurate than the ones of other machine learning algorithms. In particular, Mean Relative Error of DREAM is the smallest values between various experiments. Besides, the size of data sample DREAM uses is small, around

the size of observation windows.

9.2.3 Multi-Objective Evolutionary Algorithm

We then introduced our EMO algorithms, Non-dominated Sorting Genetic Algorithms based on Grid partitioning. There are two versions of NSGA-G using: (1) Min point and (2) Random metric. Both versions prove the advantage of our algorithms comparing to original NSGAs. The characteristic of NSGA-G promise to improve convergence, diversity and execution time in MOOPs. Besides, the way to determine the grid point is presented. This method is adapted to the number of the candidate of solutions in the last front of NSGA process.

The first NSGA-G uses the distance from the solution to the Min point to compare the quality among solutions in a group, the convergence and diversity of method are kept by randomly choosing a group to remove a solution in the last front. This strategy help algorithm executed less time more than other NSGAs. To reduce more execution time, NSGA-G with Random metric does not use the intermediate function to evaluate the value of solution in a group. Instead, the algorithm use the nature metric to remove a solution in a group in the last front. The way of choosing the nature metric is also random. Not using an intermediate function enables to reduce processing time. The convergence and the diversity are kept by the same step in NSGA-G with Min point, but the quality is reduced a little. However, it is still better than other NSGAs.

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.

9.2.4 Optimizing medical data storage configuration

We proposed Medical Data Management System (MIDAS), a DICOM management system for cloud federation. We addressed then the problem of automatically finding data storage configurations for DICOM files in a cloud federation and solved it as a MOO problem. The automatic approach [97] does not provide an approach to find an optimal data storage configuration for DICOM data storage. Our contribution is applied

NSGA-G to solve this problem to find an approximated optimal hybrid data storage configuration.

We implement MIDAS by extending the management of multiple engines platform, IReS, to organize DICOM in a cloud federation. NSGA-G is applied with DICOM extract files to find an optimal data storage configuration. The experiments show the advantage of NSGA-G among five NSGAs in execution times.

9.3 Future Works

9.3.1 Estimation

Our algorithm is developed to improve the accuracy of cost values in cloud environment. The strategy of using the coefficient of determination is applied to a multiple linear regression algorithm. Similar to Multiple Linear Regression, other machine learning algorithms need to remove expired data samples. Besides, the build model process should be as fast as possible. In addition, there is a various machine learning algorithms could be used to estimate cost values, such as Least squared regression, Bagging predictors, Multilayer Perceptron, etc. Hence, for the future, this strategy could be used to reduce the data sample and applied to these machine learning algorithms.

9.3.2 Searching and optimization

Our algorithms are proposed to improve the quality of NSGAs. They show better performances than other NSGAs. Our algorithms can be applied to MOOPs to find an approximate optimal solution. It is suitable for cloud federations when the space of candidates is huge.

In many NSGA algorithms, the size of population in each generation iterate is constant. The suitable value of population size is still a question of NSGAs. First, the genetic algorithm simulate the process of real animal population. The size of population of real world animal could decrease or increase, according to the environment (for example in case of diseases). Second, before running NSGAs, the size of population should be defined as a constant. Future works include a deeper study on the impact of the size of the population. Third, NSGAs are different in the process of removing the solution in the last front. If this process is remove, the algorithms will be faster than the original

NSGAs. As a consequence, new method of studying NSGAs is considering the size of population being variable. For example, the size of population in the next generation iterate is equal to the first front which is the first Pareto set of candidates in the ranking process.

9.3.3 Hybrid data storage configuration

The hybrid data is studied in this thesis by optimizing the data storage configuration for DICOM dataset. By using NSGA-G, the approximate optimal data storage configuration is found in the huge space of candidates. In the future, the approach can be applied to other medical data and real commercial clouds.

In the future, DREAM and NSGA-G may be applied to many other applications. For example, SparkTune [54] shows the way to estimate the cost values for Spark [135]. However, they do not consider MOOPs in their work. Hence, the estimation, searching and optimize methods for MOOPs are still the research works. For example, Spark SQL [136] generates all query execution plans. In the various resource configuration, the space of candidates is huge in MOOPs. Hence, the search and optimization need an approximate optimal solution with low complexity computation to solve MOOPs.

In addition, when the workload and data changes, the exist hybrid data configuration maybe is not the optimal solution anymore. This context raised an issue of the re-configuration. Hence, other direction of the research is the adapt method of re-configuration.

Further more, the hybrid data systems for DBMS and NoSQL is still not considered in this thesis. Hence, other direction of the research is to find a method to integrate the advantages of both DBMS and NoSQL.

PERSONAL PUBLICATIONS

- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. Dynamic estimation for medical data management in a cloud federation. *International Workshop On Data Analytics solutions for Real-Life APplications (DARLI-AP@EDBT)*, Lisbon, Portugal, 2019.
- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. Optimizing DICOM data management with NSGA-G. *International Workshop On Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP@EDBT)*, Lisbon, Portugal, 2019.
- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. An efficient multi-objective genetic algorithm for cloud computing: NSGA-G. *International workshop on on Benchmarking, Performance Tuning and Optimization for Big Data Applications (BPOD@BigData)*, Seattle, WA, USA, 2018.
- **Trung-Dung Le**, Verena Kantere, Laurent d’Orazio. Dynamic estimation and Grid partitioning approach for Multi-Objective Optimization Problems in medical cloud federations. *Submitted on Information Systems Frontiers, ISSN: 1387-3326, 2019.*

BIBLIOGRAPHY

- [1] Antonio Regalado (31 October 2011), « "Who Coined 'Cloud Computing'?" ». », *in: Technology Review. MIT* (Retrieved 31 July 2013).
- [2] Sanjay Agrawal, Vivek Narasayya, and Beverly Yang, « Integrating vertical and horizontal partitioning into automated physical database design », *in: Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04* (2004), p. 359.
- [3] Anastassia Ailamaki et al., « DBMSs on a Modern Processor: Where Does Time Go? », *in: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 266–277, ISBN: 1-55860-615-7.
- [4] M. Akdere et al., « Learning-based query performance modeling and prediction », *in: International Conference on Data Engineering* (2012), pp. 390–401.
- [5] Yahya Al-Dhuraibi et al., « Elasticity in Cloud Computing: State of the Art and Research Challenges », *in: IEEE Transactions on Services Computing* 11.2 (2018), pp. 430–447, ISSN: 19391374.
- [6] *Amazon Web Services Website*, 2018, URL: <https://aws.amazon.com/>.
- [7] *Apache Cassandra*, 2018, URL: <http://cassandra.apache.org/>.
- [8] Michael Armbrust et al., « A View of Cloud Computing », *in: Commun. ACM* 53.4 (Apr. 2010), pp. 50–58.
- [9] Michael Armbrust et al., « Above the Clouds: A Berkeley View of Cloud Computing », *in: Eecs.Berkeley.Edu* (2009).
- [10] Michael Armbrust et al., « Spark {SQL:} Relational Data Processing in Spark », *in: Proceedings of the SIGMOD International Conference on Management of Data*, Melbourne, Victoria, Australia, 2015, pp. 1383–1394.
- [11] Lucas S. Batista, « Performance Assessment of Multiobjective Evolutionary Algorithms », *in: 7* (2012).

- [12] M. A. Bayir, I. H. Toroslu, and A. Cosar, « Genetic Algorithm for the Multiple-Query Optimization Problem », *in: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.1 (2007), pp. 147–153, ISSN: 1094-6977.
- [13] Anton Beloglazov and Rajkumar Buyya, « Energy efficient allocation of virtual machines in cloud data centers », *in: CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing* (2010), pp. 577–578.
- [14] Phil Bernstein et al., « The Asilomar Report on Database Research », *in: SIGMOD Rec.* 27.4 (Dec. 1998), pp. 74–80, ISSN: 0163-5808.
- [15] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle, « An Empirical Assessment of the Properties of Inverted Generational Distance on Multi- and Many-Objective Optimization », *in: Evolutionary Multi-Criterion Optimization*, Cham: Springer International Publishing, 2017, pp. 31–45.
- [16] *BigTable*, 2018, URL: <https://cloud.google.com/bigtable/>.
- [17] Burton H. Bloom, « Space/Time Trade-offs in Hash Coding with Allowable Errors », *in: Commun. ACM* 13.7 (July 1970), pp. 422–426, ISSN: 0001-0782.
- [18] Peter A. Boncz, Stefan Manegold, and Martin L. Kersten, « Database Architecture Optimized for the New Bottleneck: Memory Access », *in: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 54–65, ISBN: 1-55860-615-7.
- [19] Peter A. Boncz, Marcin Zukowski, and Niels Nes, « MonetDB/X100: Hyper-Pipelining Query Execution », *in: CIDR 2005, Second Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA, 2005, pp. 225–237.
- [20] E. A. Boytsov, « Applying stochastic metaheuristics to the problem of data management in a multi-tenant database cluster », *in: Automatic Control and Computer Sciences* (2014).
- [21] Gilles Brassard and Paul Bratley, *Algorithmics: Theory & Practice*, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988, ISBN: 0-13-023243-2.
- [22] Leo Breiman, « Bagging predictors », *in: Machine Learning* 24 (1996), pp. 123–140.

- [23] Francesca Bugiotti et al., « Invisible Glue: Scalable Self-Tuning Multi-Stores », *in: Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, 2015.
- [24] George Candea, Neoklis Polyzotis, and Radek Vingralek, « Predictable performance and high query concurrency for data analytics », *in: VLDB Journal* 20.2 (2011), pp. 227–248, ISSN: 10668888.
- [25] V. Chankong and Y.Y. Haimes, *Multiobjective decision making: theory and methodology*, North-Holland series in system science and engineering, North Holland, 1983.
- [26] Surajit Chaudhuri, « An Overview of Query Optimization in Relational Systems », *in: Proceedings of the Symposium on Principles of Database Systems (PODS)*, Seattle, Washington, USA, 1998, pp. 34–43.
- [27] Jack Chen et al., « The MemSQL Query Optimizer: A Modern Optimizer for Real-time Analytics in a Distributed Database », *in: Proc. VLDB Endow.* 9.13 (Sept. 2016), pp. 1401–1412, ISSN: 2150-8097.
- [28] Paolo Ciaccia and Davide Martinenghi, « Reconciling Skyline and Ranking Queries », *in: Proc. VLDB Endow.* 10.11 (Aug. 2017), pp. 1454–1465, ISSN: 2150-8097.
- [29] « Cloud Federation », *in: Computing* (2011), p. 7.
- [30] Carlos A. Coello Coello and Nareli Cruz Cortés, « Solving Multiobjective Optimization Problems Using an Artificial Immune System », *in: Genetic Programming and Evolvable Machines* 6 (2005), pp. 163–190.
- [31] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, 2002.
- [32] *CouchDB*, 2018, URL: <http://couchdb.apache.org/>.
- [33] *DICOM*, 2019, URL: <https://www.dicomstandard.org/>.
- [34] Sudipto Das, Divyakant Agrawal, and Amr El Abbadi, « ElasTraS: An Elastic, Scalable, and Self-managing Transactional Database for the Cloud », *in: ACM Trans. Database Syst.* 38.1 (Apr. 2013), 5:1–5:45, ISSN: 0362-5915.

- [35] David J DeWitt et al., « Split query processing in polybase », *in: Proceedings of the SIGMOD International Conference on Management of Data*, New York, NY, USA, 2013, pp. 1255–1266.
- [36] Jeffrey Dean and Sanjay Ghemawat, « MapReduce: Simplified Data Processing on Large Clusters », *in: Commun. ACM* 51 (2008), pp. 107–113.
- [37] Kalyanmoy Deb, « Multi-objective optimization using evolutionary algorithms: an introduction », *in: KanGAL Report* (2011), pp. 1–24.
- [38] Kalyanmoy Deb and Ram Bhushan Agrawal, « Simulated Binary Crossover for Continuous Search Space », *in: Complex Systems* 9 (1994), pp. 1–34.
- [39] Kalyanmoy Deb and Himanshu Jain, « An Evolutionary Many-Objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints », *in: IEEEExplore* 18 (2013).
- [40] Kalyanmoy Deb et al., « A fast and elitist multiobjective genetic algorithm: NSGA-II », *in: IEEE Trans. Evol. Comput.* 6 (2002), pp. 182–197.
- [41] Kalyanmoy Deb et al., « Scalable Test Problems for Evolutionary Multiobjective Optimization », *in: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications* (2005), pp. 105–145.
- [42] K. Doka et al., « IReS: Intelligent, Multi-Engine Resource Scheduler for Big Data Analytics Workflows », *in: SIGMOD '15*, 2015.
- [43] Tansel Dokeroglu, Murat Ali Bayır, and Ahmet Cosar, « Integer Linear Programming Solution for the Multiple Query Optimization Problem », *in: Information Sciences and Systems 2014*, Cham: Springer International Publishing, 2014, pp. 51–60, ISBN: 978-3-319-09465-6.
- [44] Tansel Dokeroglu, Murat Ali Bayır, and Ahmet Cosar, « Robust Heuristic Algorithms for Exploiting the Common Tasks of Relational Cloud Database Queries », *in: Appl. Soft Comput.* 30.C (May 2015), pp. 72–82, ISSN: 1568-4946.
- [45] A. Elmore et al., « A Demonstration of the BigDAWG Polystore System », *in: Proc. VLDB Endow.* 8.12 (Aug. 2015), pp. 1908–1911.
- [46] Franz Färber et al., « SAP HANA Database: Data Management for Modern Business Applications », *in: SIGMOD Rec.* 40 (2012), pp. 45–51.

- [47] Franz Färber et al., « The SAP HANA Database – An Architecture Overview », *in: IEEE Data Eng. Bull.* 35 (2012), pp. 28–33.
- [48] H. M. Fard et al., « A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments », *in: 12th IEEE/ACM* (2012).
- [49] David B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (IEEE Press Series on Computational Intelligence)*, Wiley-IEEE Press, 2006, ISBN: 0471749214.
- [50] C. M. Fonseca and P. J. Fleming, « An Overview of Evolutionary Algorithms in Multiobjective Optimization », *in: Evolutionary Computation* 3.1 (1995), pp. 1–16.
- [51] Global Inter-cloud Technology Forum, « Use Cases and Functional Requirements for Inter-Cloud Computing », *in: (2010)*, p. 44.
- [52] Michael Franklin, Alon Halevy, and David Maier, « From Databases to Dataspaces: A New Abstraction for Information Management », *in: SIGMOD Rec.* 34.4 (Dec. 2005), pp. 27–33, ISSN: 0163-5808.
- [53] Chi-Wai Fung, Kamalakar Karlapalem, and Qing Li, « Cost-driven vertical class partitioning for methods in object oriented databases », *in: The VLDB Journal* 12.3 (2003), pp. 187–210.
- [54] Enrico Gallinucci and Matteo Golfarelli, « SparkTune: tuning Spark SQL through query cost modeling », *in: Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, 2019, pp. 546–549, DOI: 10.5441/002/edbt.2019.52, URL: <https://doi.org/10.5441/002/edbt.2019.52>.
- [55] A. Ganapathi et al., « Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning », *in: 2009 IEEE 25th International Conference on Data Engineering*, 2009, pp. 592–603.
- [56] Michael R. Garey and David S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, New York, NY, USA: W. H. Freeman & Co., 1990, ISBN: 0716710455.
- [57] V. Giannakouris et al., « MuSQLE: Distributed SQL query execution over multiple engine environments », *in: 2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 452–461, DOI: 10.1109/BigData.2016.7840636.

- [58] Georgios Giannikis, Gustavo Alonso, and Donald Kossmann, « SharedDB: Killing One Thousand Queries With One Stone », *in: CoRR* abs/1203.0056 (2012), arXiv: 1203.0056, URL: <http://arxiv.org/abs/1203.0056>.
- [59] Jana Giceva et al., « Deployment of Query Plans on Multicores », *in: Proc. VLDB Endow.* 8.3 (Nov. 2014), pp. 233–244, ISSN: 2150-8097.
- [60] Christian Glaßer et al., « Approximability and Hardness in Multi-objective Optimization », *in: Programs, Proofs, Processes*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 180–189.
- [61] Fred Glover and Manuel Laguna, *Tabu Search*, Norwell, MA, USA: Kluwer Academic Publishers, 1997, ISBN: 079239965X.
- [62] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989, ISBN: 0201157675.
- [63] Martin Grund et al., « HYRISE: A Main Memory Hybrid Storage Engine », *in: VLDB Endow.* 4 (2010), pp. 105–116.
- [64] *HBase*, 2018, URL: <https://hbase.apache.org/>.
- [65] Richard A. Hankins and Jignesh M. Patel, « Data Morphing: An Adaptive, Cache-conscious Storage Technique », *in: Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, VLDB Endowment, 2003, pp. 417–428.
- [66] Stavros Harizopoulos, Daniel J. Abadi, and Samuel Madden, « Performance Tradeoffs in Read-Optimized Databases », *in: VLDB* (2006), pp. 487–498.
- [67] Florian Helff and Laurent Orazio, « Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments », *in: EDBT/ICDT Workshops*, 2016.
- [68] Frederick S. Hillier and Gerald J. Lieberman, *Introduction to Operations Research, 4th Ed.* San Francisco, CA, USA: Holden-Day, Inc., 1986, ISBN: 0816238715.
- [69] S. Huband et al., « A review of multiobjective test problems and a scalable test problem toolkit », *in: IEEE Transactions on Evolutionary Computation* 10.5 (2006), pp. 477–506, ISSN: 1089-778X, DOI: 10.1109/TEVC.2005.861417.
- [70] *HyperTable*, 2018, URL: <http://www.hypertable.org/>.

- [71] H. Ishibuchi, H. Masuda, and Y. Nojima, « Sensitivity of performance evaluation results by inverted generational distance to reference points », *in: 2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1107–1114.
- [72] Himanshu Jain and Kalyanmoy Deb, « 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* 18 (2014), pp. 602–622.
- [73] Manos Karpathiotakis, Ioannis Alagiannis, and Anastasia Ailamaki, « Fast Queries over Heterogeneous Data Through Engine Customization », *in: Proc. VLDB Endow.* 9.12 (Aug. 2016), pp. 972–983, ISSN: 2150-8097.
- [74] G. Keller, *Statistics for Management and Economics*, Cengage Learning, 2014, ISBN: 9781133420774.
- [75] Salman A. Khan and Shafiqur Rehman, « Iterative non-deterministic algorithms in on-shore wind farm design: A brief survey », *in: Renewable and Sustainable Energy Reviews* 19 (2013), pp. 370 –384.
- [76] V. Khare, X. Yao, and K. Deb, « Performance Scaling of Multi-objective Evolutionary Algorithms », *in: Evolutionary Multi-Criterion Optimization*, Berlin, Heidelberg, 2003, pp. 376–390.
- [77] Alireza Khoshkbarforousha et al., « Flower: A Data Analytics Flow Elasticity Manager », *in: PVLDB* 10 (2017), pp. 1893–1896.
- [78] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, « Neurocomputing: Foundations of Research », *in: ed. by James A. Anderson and Edward Rosenfeld*, Cambridge, MA, USA: MIT Press, 1988, chap. Optimization by Simulated Annealing, pp. 551–567, ISBN: 0-262-01097-6, URL: <http://dl.acm.org/citation.cfm?id=65669.104435>.
- [79] Herald Killapi et al., « Schedule optimization for data processing flows on the cloud », *in: Proceedings of the 2011 international conference on Management of data - SIGMOD '11* (2011), p. 289.
- [80] Konstantinos Kloudas et al., « Pixida: Optimizing Data Parallel Jobs in Wide-Area Data Analytics », *in: Vldb* (2014), pp. 72–83, ISSN: 2150-8097.

- [81] J. Knowles and D. Corne, « The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation », *in: 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 1, 1999, pp. 98–105.
- [82] Boyan Kolev et al., « CloudMdsQL: querying heterogeneous cloud data stores with a common language », *in: Distributed and Parallel Databases 34.4* (2016), pp. 463–503.
- [83] Boyan Kolev et al., « The CloudMdsQL Multistore System », *in: Proceedings of the SIGMOD International Conference on Management of Data*, San Francisco, CA, USA, 2016, pp. 2113–2116.
- [84] Mario Köppen and Kaori Yoshida, « Substitute Distance Assignments in NSGA-II for Handling Many-objective Optimization Problems », *in: (2006)*, pp. 727–741.
- [85] T. V. V. Kumar and K. Devi, « Frequent queries identification for constructing materialized views », *in: 2011 3rd International Conference on Electronics Computer Technology*, vol. 6, 2011, pp. 177–181.
- [86] Jeff LeFevre et al., « {MISO:} souping up big data query processing with a multistore system », *in: Proceedings of the SIGMOD International Conference on Management of Data*, Snowbird, UT, USA, 2014, pp. 1591–1602.
- [87] Jongwuk Lee and Seung won Hwang, « Toward efficient multidimensional subspace skyline computation », *in: VLDB Journal 23.1* (2014), pp. 129–145, ISSN: 10668888.
- [88] Viktor Leis et al., « How Good Are Query Optimizers, Really? », *in: Proc. VLDB Endow. 9.3* (Nov. 2015), pp. 204–215, ISSN: 2150-8097.
- [89] *LevelDB*, 2018, URL: <http://leveldb.org/>.
- [90] Jiexing Li, Jeffrey F Naughton, and Rimma V Nehme, « Resource bricolage and resource selection for parallel database systems », *in: VLDBJ 26.1* (2017), pp. 31–54.
- [91] Harold Lim, Y Han, and S Babu, « How to Fit when No One Size Fits. », *in: Cidr* (2013).
- [92] Ji Liu et al., « Multi-objective scheduling of Scientific Workflows in multisite clouds », *in: Future Generation Computer Systems 63* (2016), pp. 76–95.

- [93] Benjamin Markines et al., « Evaluating Similarity Measures for Emergent Semantics of Social Tagging », *in: Proceedings of the 18th International Conference on World Wide Web, WWW '09*, Madrid, Spain: ACM, 2009, pp. 641–650, ISBN: 978-1-60558-487-4.
- [94] Zbigniew Michalewicz, *How to Solve It: Modern Heuristics 2e*, Berlin, Heidelberg: Springer-Verlag, 2010, ISBN: 3642061346, 9783642061349.
- [95] *Microsoft Azure Website*, 2018, URL: <https://azure.microsoft.com/>.
- [96] *MongoDB*, 2018, URL: <https://www.mongodb.com/>.
- [97] Cong-Danh NGUYEN, « Workload- and Data-based Automated Design for a Hybrid Row-column Storage Model and Bloom Filter-based Query Processing for Large-scale DICOM Data Management », PhD thesis, Clermont Auvergne University, 2018.
- [98] Arnab Nandi and H V Jagadish, « Guided Interaction: Rethinking the Query-Result Paradigm », *in: VLDB 4.12* (2011), pp. 1466–1469.
- [99] Richard E. Neapolitan and Kumarss Naimipour, *Foundations of Algorithms*, Lexington, MA, USA: D. C. Heath and Company, 1996, ISBN: 0-669-35298-5.
- [100] *Neo4J*, 2018, URL: <https://neo4j.com/>.
- [101] Danh Nguyen-Cong et al., « Storing and Querying DICOM Data with HYTORMO », *in: Data Management and Analytics for Medicine and Healthcare*, Cham: Springer International Publishing, 2017, pp. 43–61.
- [102] Hiep Nguyen et al., « AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service », *in: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, San Jose, CA: USENIX, 2013, pp. 69–82, ISBN: 978-1-931971-02-7.
- [103] T. Nykiel et al., « MRShare: sharing across multiple queries in MapReduce », *in: VLDB Endowment* (2010).
- [104] Frank J. Ohlhorst, *Big Data Analytics: Turning Big Data into Big Money*, 1st, Wiley Publishing, 2012, ISBN: 1118147596, 9781118147597.
- [105] An Oracle and White Paper, « Performance Evaluation of Storage and Retrieval of DICOM Image Content in Oracle Database 11g Using HP Blade Servers and Intel Processors », *in: January* (2010).

- [106] Andrzej Osyczka, *Multicriterion optimization in engineering with FORTRAN programs*, Chichester, 1984, ISBN: 0853124817.
- [107] M. Tamer Özsu and Patrick Valduriez, *Principles of distributed database systems, third edition*, 2011.
- [108] S. Papadomanolakis and A. Ailamaki, « AutoPart: automating schema design for large scientific databases using data partitioning », *in: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004*. 2004, pp. 383–392.
- [109] Yannis Papakonstantinou, « Polystore Query Rewriting: The Challenges of Variety », *in: EDBT/ICDT Workshops*, 2016.
- [110] Yoonjae Park, Jun-Ki Min, and Kyuseok Shim, « Processing of Probabilistic Skyline Queries Using MapReduce », *in: Proc. VLDB Endow.* 8.12 (Aug. 2015), pp. 1406–1417, ISSN: 2150-8097.
- [111] Judea Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984, ISBN: 0-201-05594-5.
- [112] Fabian Pedregosa et al., « Scikit-learn: Machine Learning in Python », *in: J. Mach. Learn. Res.* 12 (Nov. 2011), pp. 2825–2830, ISSN: 1532-4435.
- [113] S. Raschka, *Python Machine Learning: Unlock Deeper Insights Into Machine Learning*, Community experience distilled, Packt Publishing, 2015, ISBN: 9781783555130.
- [114] *Riak*, 2018, URL: <https://riak.com/index.html>.
- [115] Peter J. Rousseeuw and Annick M. Leroy, *Robust regression and outlier detection*, 1987.
- [116] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd, Upper Saddle River, NJ, USA: Prentice Hall Press, 2009, ISBN: 0136042597, 9780136042594.
- [117] Hans-Paul Paul Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*, New York, NY, USA: John Wiley & Sons, Inc., 1993, ISBN: 0471571482.

- [118] Haitham Seada, Mohamed Abouhawwash, and Kalyanmoy Deb, « Towards a Better Balance of Diversity and Convergence in NSGA-III: First Results », *in: Evolutionary Multi-Criterion Optimization*, Cham: Springer International Publishing, 2017, pp. 545–559.
- [119] Juwei Shi et al., « Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics », *in: Proc. VLDB Endow.* 8.13 (Sept. 2015), pp. 2110–2121, ISSN: 2150-8097.
- [120] S. Sidhanta, W. Golab, and S. Mukhopadhyay, « OptEx: A Deadline-Aware Cost Optimization Model for Spark », *in: IEEE/ACM* (2016).
- [121] Tsu T. Soong, *Fundamentals of probability and statistics for engineers*, John Wiley & Sons, 2004.
- [122] N. Srinivas and K. Deb, « Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms », *in: Evolutionary Computation 2* (1994), pp. 221–248.
- [123] Michael Stonebraker and Uğur Çetintemel, « "One Size Fits All": An Idea Whose Time Has Come and Gone », *in: Proceedings of the 21st International Conference on Data Engineering, ICDE '05*, Washington, DC, USA: IEEE Computer Society, 2005, pp. 2–11, ISBN: 0-7695-2285-8, DOI: 10.1109/ICDE.2005.1, URL: <https://doi.org/10.1109/ICDE.2005.1>.
- [124] Michael Stonebraker et al., « The End of an Architectural Era: (It's Time for a Complete Rewrite) », *in: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07*, Vienna, Austria: VLDB Endowment, 2007, pp. 1150–1160, ISBN: 978-1-59593-649-3.
- [125] Mike Stonebraker et al., « C-store: A Column-oriented DBMS », *in: International Conference on Very Large Data Bases (VLDB '05)*, Trondheim, Norway, 2005, pp. 553–564.
- [126] Er Strehl and Joydeep Ghosh, « Value-based Customer Grouping from Large Retail Data-sets », *in: Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery, Orlando* (Mar. 2000).
- [127] *The Galactica Website*, 2018, URL: <https://horizon.isima.fr>.
- [128] *The HDFS Website*, 2018, URL: <http://hadoop.apache.org/>.

- [129] *The Hive Website*, 2018, URL: <http://hive.apache.org/>.
- [130] *The MOEA Website*, 2018, URL: <http://moeaframework.org/>.
- [131] *The MemSQL Website*, 2018, URL: <https://www.memsql.com/>.
- [132] *The Oracle Website*, 2018, URL: <https://www.oracle.com/>.
- [133] *The PostgreSQL Website*, 2018, URL: <https://www.postgresql.org/>.
- [134] *The SQL Website*, 2018, URL: <https://www.mysql.com/>.
- [135] *The Spark Website*, 2018, URL: <https://spark.apache.org/>.
- [136] *The SparkSQL Website*, 2018, URL: <https://spark.apache.org/sql/>.
- [137] *The TPC-H Website*, 2018, URL: <http://www.tpc.org/tpch/>.
- [138] *The Weka Website*, 2018, URL: <https://www.cs.waikato.ac.nz/ml/weka/>.
- [139] Ashish Thusoo et al., « Hive - {A} Warehousing Solution Over a Map-Reduce Framework », in: *Proceedings of the Very Large Data Bases Endowment (PVLDB) 2.2* (2009), pp. 1626–1629.
- [140] Ashish Thusoo et al., « Hive - a petabyte scale data warehouse using Hadoop », in: *Proceedings of the International Conference on Data Engineering (ICDE)*, Long Beach, California, {USA}, 2010, pp. 996–1005.
- [141] Sean Tozer, Tim Brecht, and Ashraf Aboulnaga, « Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads », in: *International Conference on Data Engineering* (2010), pp. 397–408.
- [142] Immanuel Trummer and Christoph Koch, « A Fast Randomized Algorithm for Multi-Objective Query Optimization », in: (2016).
- [143] Immanuel Trummer and Christoph Koch, « Approximation Schemes for Many-objective Query Optimization », in: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14*, Snowbird, Utah, USA: ACM, 2014, pp. 1299–1310, ISBN: 978-1-4503-2376-5.
- [144] Immanuel Trummer and Christoph Koch, « Multi-objective parametric query optimization », in: *VLDB J.* 8 (2016).
- [145] Immanuel Trummer and Christoph Koch, « Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer », in: *Proc. VLDB Endow.* 9.9 (May 2016), pp. 648–659, ISSN: 2150-8097.

- [146] Prasang Upadhyaya, Magdalena Balazinska, and Dan Suciu, « How to Price Shared Optimizations in the Cloud », in: *Proceedings of the VLDB Endowment* 5 (Feb. 2012).
- [147] Dana Van Aken et al., « Automatic Database Management System Tuning Through Large-scale Machine Learning », in: *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD '17*, Chicago, Illinois, USA: ACM, 2017, pp. 1009–1024, ISBN: 978-1-4503-4197-4.
- [148] David A. Van Veldhuizen and Gary B. Lamont, « Evolutionary Computation and Convergence to a Pareto Front », in: *Late Breaking Papers at the Genetic Programming 1998 Conference* (1998), pp. 221–228.
- [149] J. Veiga et al., « Performance evaluation of big data frameworks for large-scale data analytics », in: *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 424–431.
- [150] David A. Van Veldhuizen and David A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, tech. rep., Evolutionary Computation, 1999.
- [151] Alessandro Vicini et al., « Multipoint transonic airfoil design by means of a multi-objective genetic algorithm », in: *35th Aerospace Sciences Meeting and Exhibit*, 1997, p. 82.
- [152] Wendy Wolfson, « MapReduce: Simplified Data Processing on Large Clusters », in: *Chemistry and Biology* 19.9 (2012), pp. 1075–1076, ISSN: 10745521.
- [153] W. Wu et al., « Predicting query execution time: Are optimizer cost models really unusable? », in: *IEEE 29th International Conference on Data Engineering (ICDE)*, 2013.
- [154] Pengcheng Xiong, Ferst Drive, and Yun Chi, « ActiveSLA : A Profit-Oriented Admission Control Framework for Database-as-a-Service Providers Categories and Subject Descriptors », in: *2nd ACM Symposium on Cloud Computing SOCC 11* (2011), pp. 1–14.
- [155] G.G. Yen and Z. He, « Performance Metrics Ensemble for Multiobjective Evolutionary Algorithms », in: *IEEE Transactions on Evolutionary Computation* (2013).

- [156] Wenhui Yu et al., « Fast Algorithms for Pareto Optimal Group-based Skyline », *in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, Singapore, Singapore: ACM, 2017, pp. 417–426, ISBN: 978-1-4503-4918-5.
- [157] Steffen Zeuch, Holger Pirk, and Johann-Christoph Freytag, « Non-invasive progressive optimization for in-memory databases », *in: Proceedings of the VLDB Endowment 9.14* (2016), pp. 1659–1670, ISSN: 21508097.
- [158] Q. Zhang and H. Li, « MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition », *in: IEEE Transactions on Evolutionary Computation 11* (2007), pp. 712–731.
- [159] Jianqiao Zhu et al., « Looking ahead makes query plans robust », *in: Proceedings of the VLDB Endowment 10.8* (2017), pp. 889–900, ISSN: 21508097.
- [160] E. Zitzler et al., « Performance assessment of multiobjective optimizers: an analysis and review », *in: IEEE Transactions on Evolutionary Computation 7* (2003), pp. 117–132.
- [161] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, « SPEA2: Improving the strength Pareto evolutionary algorithm », *in: TIK-report 103* (2001).
- [162] M. A. et al., « TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems », *in: CoRR, abs/1603.04467* (Jan. 2016).

Titre: Gestion de masses de données dans une fédération de nuages informatiques

Mot clés : La régression linéaire multiple; Fédérations de nuages; Optimisation multi-objectifs; Solutions pareto-optimales; Algorithmes génétiques; Algorithme de tri génétique non dominé.

Resumé : Les fédérations de nuages informatiques peuvent être considérées comme une avancée majeure dans l'informatique en nuage, en particulier dans le domaine médical. En effet, le partage de données médicales améliorerait la qualité des soins. La fédération de ressources permettrait d'accéder à toutes les informations, même sur une personne mobile, avec des données hospitalières distribuées sur plusieurs sites. En outre, cela permettrait d'envisager de plus grands volumes de données sur plus de patients et ainsi de fournir des statistiques plus fines.

Les données médicales sont généralement conformes à la norme DICOM (Digital Imaging and Communications in Medicine). Les fichiers DICOM peuvent être stockés sur différentes plates-formes, telles qu'Amazon, Microsoft, Google Cloud, etc. La gestion des fichiers, y compris le partage et le traitement, sur ces

plates-formes, suit un modèle de paiement à l'utilisation, selon des modèles de prix distincts et en s'appuyant sur divers systèmes de gestion de données (systèmes de gestion de données relationnelles ou SGBD ou systèmes NoSQL). En outre, les données DICOM peuvent être structurées en lignes ou colonnes ou selon une approche hybride (ligne-colonne). En conséquence, la gestion des données médicales dans des fédérations de nuages soulève des problèmes d'optimisation multi-objectifs (MOOP - Multi-Objective Optimization Problems) pour (1) le traitement des requêtes et (2) le stockage des données, selon les préférences des utilisateurs, telles que le temps de réponse, le coût monétaire, la qualité, etc. Ces problèmes sont complexes à traiter en raison de la variabilité de l'environnement (liée à la virtualisation, aux communications à grande échelle, etc.).

Pour résoudre ces problèmes, nous

proposons MIDAS (Medical system on cloud federAtionS), un système médical sur les fédérations de groupes. Premièrement, MIDAS étend IReS, une plateforme open source pour la gestion de flux de travaux d'analyse sur des environnements avec différents systèmes de gestion de bases de données. Deuxièmement, nous proposons un algorithme d'estimation des valeurs de coût dans une fédération de nuages, appelé Algorithme de régression dynamique (DREAM). Cette approche permet de s'adapter à la variabilité de l'environnement en modifiant la taille des données à des fins de formation et de test, et d'éviter d'utiliser des informations expirées sur les systèmes. Troisièmement, l'algorithme génétique de tri non dominé à base de grilles (NSGA-G) est proposé pour résoudre des problèmes d'optimisation multi-critères en présence d'espaces de candidats de grande taille. NSGA-G vise à trouver une solution op-

timale approximative, tout en améliorant la qualité du front de Pareto. En plus du traitement des requêtes, nous proposons d'utiliser NSGA-G pour trouver une solution optimale approximative à la configuration de données DICOM.

Nous fournissons des évaluations expérimentales pour valider DREAM, NSGA-G avec divers problèmes de test et jeux de données. DREAM est comparé à d'autres algorithmes d'apprentissage automatique en fournissant des coûts estimés précis. La qualité de la NSGA-G est comparée à celle des autres algorithmes NSGA présentant de nombreux problèmes dans le cadre du MOEA. Un jeu de données DICOM est également expérimenté avec NSGA-G pour trouver des solutions optimales. Les résultats expérimentaux montrent les qualités de nos solutions en termes d'estimation et d'optimisation de problèmes multi-objectifs dans une fédération de nuages.

Title: Data Management in a Cloud Federation

Keywords : Multiple Linear Regression; Cloud federations; Multi-Objective Optimization; Pareto-optimal solutions, Genetic algorithms; Non-dominated Sorting Genetic Algorithm.

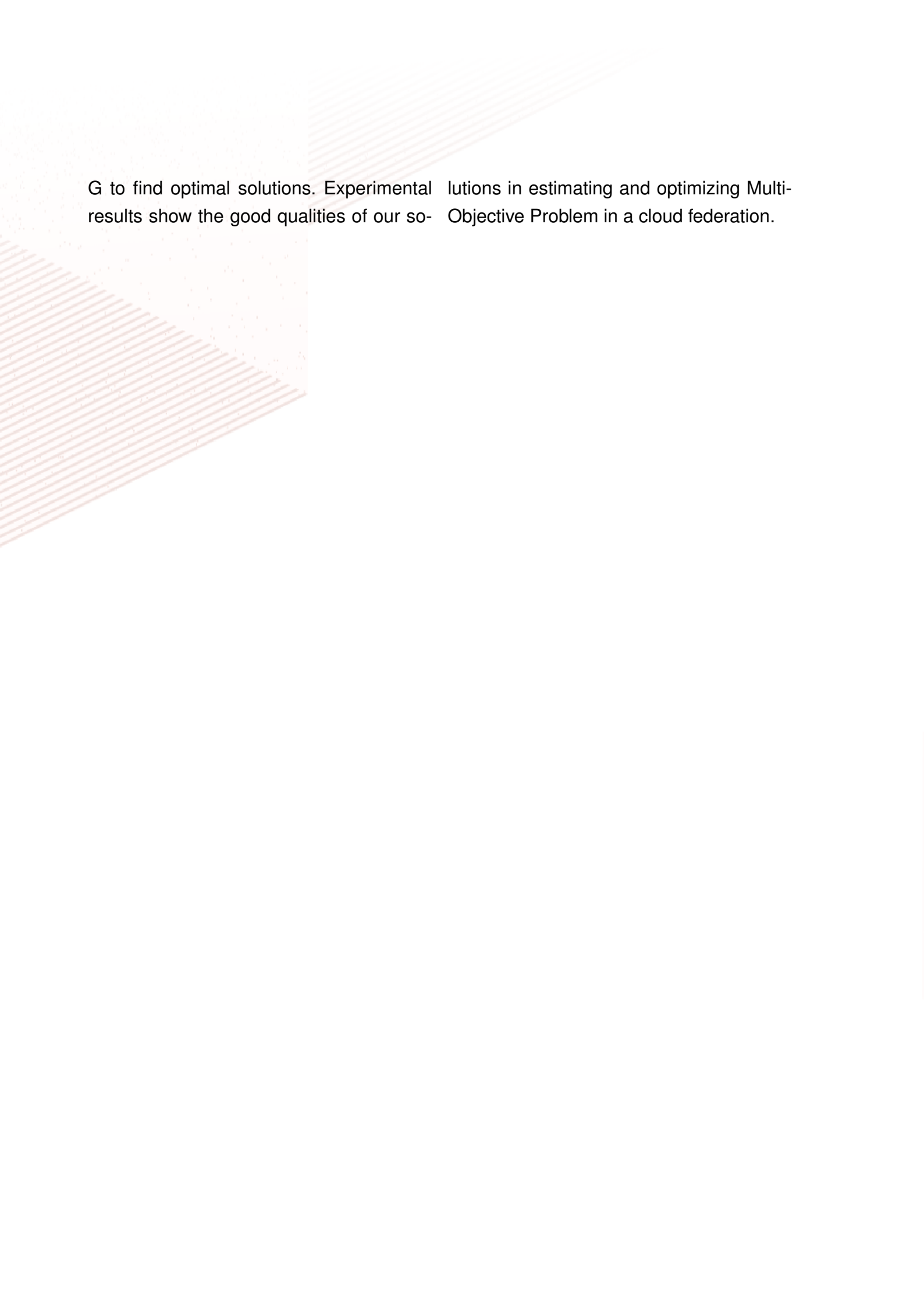
Abstract : Cloud federations can be seen in particular in the medical domain. In as major progress in cloud computing, indeed, sharing medical data would improve

healthcare. Federating resources makes it possible to access any information even on a mobile person with distributed hospital data on several sites. Besides, it enables us to consider larger volumes of data on more patients and thus provide finer statistics.

Medical data usually conform to the Digital Imaging and Communications in Medicine (DICOM) standard. DICOM files can be stored on different platforms, such as Amazon, Microsoft, Google Cloud, etc. The management of the files, including sharing and processing, on such platforms, follows the pay-as-you-go model, according to distinct pricing models and relying on various systems (Relational Data Management Systems or DBMSs or NoSQL systems). In addition, DICOM data can be structured following traditional (row or column) or hybrid (row-column) data storages. As a consequence, medical data management in cloud federations raises Multi-Objective Optimization Problems (MOOPs) for (1) query processing and (2) data storage, according to users preferences, related to various measures, such as response time, monetary cost, qualities, etc. These problems are complex to address because of heterogeneous database engines, the variability (due to virtualization, large-scale communications, etc.) and high computational complexity of a cloud federation.

To solve these problems, we propose a Medical system on cloud federations (MIDAS). First, MIDAS extends IReS, an open source platform for complex analytics workflows executed over multi-engine environments, to solve MOOP in the heterogeneous database engines. Second, we propose an algorithm for estimating of cost values in a cloud environment, called Dynamic REgression Algorithm (DREAM). This approach adapts the variability of cloud environment by changing the size of data for training and testing process to avoid using the expire information of systems. Third, Non-dominated Sorting Genetic Algorithm based on Grid partitioning (NSGA-G) is proposed to solve the problem of MOOP is that the candidate space is large. NSGA-G aims to find an approximate optimal solution, while improving the quality of the optimal Pareto set of MOOP. In addition to query processing, we propose to use NSGA-G to find an approximate optimal solution for DICOM data configuration.

We provide experimental evaluations to validate DREAM, NSGA-G with various test problem and dataset. DREAM is compared with other machine learning algorithms in providing accurate estimated costs. The quality of NSGA-G is compared to other NSGAs with many problems in MOEA framework. The DICOM dataset is also experimented with NSGA-



G to find optimal solutions. Experimental results show the good qualities of our solutions in estimating and optimizing Multi-Objective Problem in a cloud federation.