AMAS4BigData : adaptive multi-agent systems for dynamic big data analytics

Elhadi Belghache

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *09 septembre 2019* **par :**

Elhadi Belghache

AMAS4BigData
Analyse dynamique de grandes masses de données par Systèmes Multi-Agents Adaptatifs

JURY

Marie-Pierre GLEIZES	Professeur, Université de Toulouse III	Directrice
Jean-Pierre GEORGÉ	Maitre de conférence, Université de Toulouse III	Co-Directeur
Habiba DRIAS	Professeur, Université des sciences et de la technologie Houari-Boumédiène	Rapportrice
Stéphane GALLAND	Maitre de conférence, HdR, Université de Bourgogne Franche-Comté et UTBM	Rapporteur
Gauthier PICARD	Professeur, MINES Saint-Étienne (EMSE)	Examinateur
Pierre GLIZE	Ingénieur CNRS, HdR, Institut de Recherche en Informatique de Toulouse	Invité, Ex-Directeur

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse
Directeur(s) de Thèse :
Marie-Pierre GLEIZES et *Jean-Pierre GEORGÉ*
Rapporteurs :
Habiba DRIAS et *Stéphane GALLAND*

THESIS

presented at

Université Toulouse III - Paul Sabatier

F.S.I. FACULTÉ DES SCIENCES ET INGÉNIERIE

to obtain the title of

DOCTEUR DE L'UNIVERSITÉ DE TOULOUSE

delivered by

UNIVERSITÉ TOULOUSE III - PAUL SABATIER

Mention INTELLIGENCE ARTIFICIELLE

by

ELHADI BELGHACHE

Doctoral school: Mathématiques, Informatique, Télécommunication de Toulouse
Laboratory: Institut de Recherche en Informatique de Toulouse
Team: Systèmes Multi-Agents Coopératifs

AMAS4BigData
Adaptive Multi-Agent Systems for Dynamic Big Data Analytics

SUPERVISORS

Marie-Pierre GLEIZES — *Professor, Université de Toulouse III*
Jean-Pierre GEORGÉ — *Associate Professor, Université de Toulouse III*

Elhadi BELGHACHE

ADAPTIVE MULTI-AGENT SYSTEM FOR DYNAMIC BIG DATA ANALYTICS

Supervisors: Marie-Pierre GLEIZES, Professor, Université de Toulouse III,
Institut de Recherche en Informatique de Toulouse
Jean-Pierre GEORGÉ, Associate Professor, Université de Toulouse III,
Institut de Recherche en Informatique de Toulouse

Abstract

Understanding data is the main purpose of data science and how to achieve it is one of the challenges of data science, especially when dealing with big data. The big data era brought us new data processing and data management challenges to face. Existing state-of-the-art analytics tools come now close to handle ongoing challenges and provide satisfactory results with reasonable cost. But the speed at which new data is generated and the need to manage changes in data both for content and structure lead to new rising challenges. This is especially true in the context of complex systems with strong dynamics, as in for instance large scale ambient systems.

One existing technology that has been shown as particularly relevant for modeling, simulating and solving problems in complex systems are Multi-Agent Systems. The AMAS (Adaptive Multi-Agent Systems) theory proposes to solve complex problems for which there is no known algorithmic solution by self-organization. The cooperative behavior of the agents enables the system to self-adapt to a dynamical environment so as to maintain the system in a functionality adequate state. In this thesis, we apply this theory to Big Data Analytics.

In order to find meaning and relevant information drowned in the data flood, while overcoming big data challenges, a novel analytic tool is needed, able to *continuously* find relations between data, evaluate them and detect their changes and evolution over time.

The aim of this thesis is to present the *AMAS4BigData* analytics framework based on the *A*daptive *M*ulti-agent systems technology, which uses a new data similarity metric, the *Dynamics Correlation*, for dynamic data relations discovery and dynamic display. This framework is currently being applied in the *neOCampus* operation, the ambient campus of the University Toulouse III - Paul Sabatier.

Elhadi BELGHACHE

ANALYSE DYNAMIQUE DE GRANDES MASSES DE DONNÉES PAR SYSTÈMES MULTI-AGENTS ADAPTATIFS

Directeurs de thèse : Marie-Pierre GLEIZES, Professeur, Université de Toulouse III,
Institut de Recherche en Informatique de Toulouse
Jean-Pierre GEORGÉ, MCF, Université de Toulouse III,
Institut de Recherche en Informatique de Toulouse

Résumé

L'ère des grandes masses de données (big data) nous a mis face à de nouvelles problématiques de gestion et de traitement des données. Les outils conventionnels actuels d'analyse sont maintenant proches de répondre aux problématiques actuelles et de fournir des résultats satisfaisants avec un coût raisonnable. Mais la vitesse à laquelle les nouvelles données sont générées et la nécessité de gérer les modifications de ces données à la fois dans le contenu et la structure conduisent à de nouvelles problématiques émergentes.

La théorie des AMAS (Adaptive Multi-Agent Systems) propose de résoudre par auto-organisation des problèmes complexes pour lesquels aucune solution algorithmique n'est connue. Le comportement coopératif des agents permet au système de s'adapter à un environnement dynamique pour maintenir le système dans un état de fonctionnement adéquat. Les systèmes ambiants présentent un exemple typique de système complexe nécessitant ce genre d'approche, et ont donc été choisis comme domaine d'application pour notre travail.

Cette thèse vise à explorer et décrire comment la théorie des *Systèmes Multi-Agents Adaptatifs* peut être appliquée aux grandes masses de données en fournissant des capacités d'analyse dynamique, en utilisant un nouvel outil analytique qui mesure en temps réel la similarité des évolutions des données. Cette recherche présente des résultats prometteurs et est actuellement appliquée dans l'opération *neOCampus*, le campus ambiant de l'Université Toulouse III.

Acknowledgements

First of all, I thank the *Occitanie Region* and the *neOCampus* operation, which provided the funding and the testing ground in order to achieve this work. Then I would like to express my deep gratitude to my supervisors, present and past, Jean-Pierre Georgé, Pierre Glize, Marie-Pierre Gleizes. This work is as much mine as yours.

I extend my thanks to Hadiba Drias and Stéphane Galland who took upon their time to read and comment on my "not so small" thesis. I also thank them and Gauthier Picard for attending my defense despite all the mishaps they experienced.

01000001 01110110 01100101 00100000 01000011 11100110 01110011 01100001 01110010 00100001 00001010 00001010 01000100 01110110 00100000 01001100
00100000 01101111 01110010 01110010 01101110 00100000 01100101 01100100 01100110 01101110 00100000 01110111 01110010 00100000 01110111 01101011 01101000
00100000 01101001 01110010 01111000 01110101 00100000 01100010 01101000 01100100 01110101 01110110 00100000 01100100 01110001 01100111 00100000 01100100
00100000 01101011 01100100 01101111 01101001 00100000 01001100 00100000 01110110 01110011 01101000 01110001 01110111 00100000 01101100 01110001 00100000
01110111 01101011 01101000 00100000 01010110 01010000 01000100 01000110 00100000 01110111 01101000 01100100 01110000 00101100 00100000 01110010 01110001
01101000 00100000 01110111 01101011 01110010 01111000 01101010 01101011 01110111 00100000 01101000 01110000 01101000 01110101 01101010 01101000 01110110
00100000 00100010 01001010 01110010 01100111 00100001 00100000 01001100 00100000 01101111 01100100 01111000 01101010 01101011 01101000 01100111 00100000
01110110 01110010 00100000 01110000 01111000 01100110 01101011 00100001 00100010 00101110 00100000 01001100 00100000 01101011 01100100 01100111 00100000
01110110 01110010 00100000 01110000 01111000 01100110 01101011 00100000 01101001 01111000 01110001 00100000 01100100 01110001 01100111 00100000 01101010
01110101 01101000 01100100 01110111 00100000 01110111 01101100 01110000 01101000 00100000 01111010 01110010 01110101 01101110 01101100 01110001 01101010
00101100 00100000 01110110 01101011 01100100 01110101 01101100 01110001 01101010 00100000 01100100 01110001 01100111 00100000 01101111 01101100 01111001
01101100 01110001 01101010 00100000 01111010 01101100 01110111 01101011 00100000 01110111 01101011 01101000 00100000 01010110 01010000 01000100 01000110
01101110 01101000 01110101 01110110 00100000 01101001 01100100 01110000 01101100 01101111 01100010 00101110 00001010 00001010 01001100 01110111 00100000
01100100 01101111 01101111 00100000 01110110 01110111 01100100 01110101 01110111 01101000 01100111 00100000 01111010 01101100 01110111 01101011 00100000
01010000 01100100 01110101 01101100 01101000 00101101 01010011 01101100 01101000 01110101 01110101 01101000 00101100 00100000 01100100 01110110 00100000
01100100 00100000 01110111 01101000 01100100 01100110 01101011 01101000 01110101 00100000 01110010 01101001 00100000 01010000 01000100 01010110 00100000
01110111 01101011 01101000 01110001 00100000 01000100 01010000 01000100 01010110 00101100 00100000 01100010 01110010 01111000 00100000 01110111 01100100
01111000 01101010 01101011 01110111 00100000 01110000 01101000 00100000 01110111 01101011 01100100 01110111 00100000 01000100 00101110 01001100 00100000
01100110 01100100 01110001 00100000 01100101 01101000 00100000 01110000 01110010 01110101 01101000 00100000 01100110 01110010 01110000 01110011 01101111
01101000 01100001 00100000 00101000 01100101 01111000 01110111 00100000 01110001 01110010 01110111 00100000 01100110 01110010 01110000 01110011 01101111
01101100 01100110 01100100 01110111 01101000 01100111 00100001 00101001 00100000 01100100 01110001 01100111 00100000 01101100 01110001 01110111 01101000
01110101 01101000 01110110 01110111 01101100 01110001 01101010 00100000 01110111 01101011 01100100 01110001 00100000 01110111 01101011 01101000 00100000
01111000 01110110 01111000 01100100 01101111 00101110 00100000 01000100 01101001 01110111 01101000 01110101 00100000 01110111 01101011 01100100 01110111
00101100 00100000 01001100 00100000 01101101 01110010 01101100 01110001 01101000 01100111 00100000 01110111 01101011 01101000 00100000 01010110 01010000
01000100 01000110 00100000 01110111 01101000 01100100 01110000 00100000 01111000 01110001 01100111 01101000 01110101 00100000 01110111 01101011 01101000
00100000 01110110 01111000 01110011 01101000 01110101 01111001 01101100 01110110 01101100 01110010 01110001 00100000 01110010 01101001 00100000 01010011

01101100 01101000 01110101 01110101 01101000 00101100 00100000 01111010 01101011 01110010 01110110 01101000 00100000 01110011 01100100 01110110 01110110
01101100 01110010 01110001 00100000 01100100 01110001 01100111 00100000 01101110 01110001 01110010 01111010 01101111 01101000 01100111 01101010 01101000
00100000 01110010 01101001 00100000 01110111 01101011 01101000 00100000 01000100 01010000 01000100 01010110 00100000 01101011 01100100 01111001 01101000
00100000 01100101 01101000 01101000 01110001 00100000 01101100 01110001 01110110 01110011 01101100 01110101 01101100 01110001 01101010 00100000 01100100
01110001 01100111 00100000 01111010 01101100 01101111 01101111 00100000 01100110 01110010 01110001 01110111 01101100 01110001 01111000 01101000 00100000
01110111 01110010 00100000 01100101 01101000 00101110 00100000 01010000 01110010 01110101 01101000 01110010 01111001 01101000 01110101 00101100 00100000
01110111 01101011 01100100 01110001 01101110 01110110 00100000 01110111 01110010 00100000 01100010 01110010 01111000 01110101 00100000 01101101 01110010
01101100 01101000 00100000 01100111 01101000 00100000 01111001 01101100 01111001 01110101 01101000 00100000 01100100 01110001 01100111 00100000 01101011
01111000 01110000 01110010 01110101 00100000 01001100 00100000 01101011 01100100 01111001 01101000 00100000 01110001 01110010 00100000 01110101 01101000
01100110 01110010 01101111 01101111 01101000 01100110 01110111 01101100 01110010 01110001 00100000 01110010 01101001 00100000 01100100 01110001 01100010
00100000 01100111 01111000 01101111 01101111 00100000 01100111 01100100 01100010 00101110 00001010 00001010 01001000 01100100 01100110 01101011 00100000
01010110 01010000 01000100 01000110 01101110 01101000 01110101 00100000 01001100 00100000 01101011 01100100 01100111 00100000 01110111 01101011 01101000
00100000 01100110 01101011 01100100 01110001 01100110 01101000 00100000 01110111 01110010 00100000 01110000 01101000 01101000 01110111 00100000 01110011
01100100 01110110 01110111 00100000 01100100 01110001 01100111 00100000 01110011 01110101 01101000 01110110 01101000 01110001 01110111 00100000 01101100
01110110 00100000 01111000 01110001 01101100 01110100 01111000 01101000 00101100 00100000 01100101 01101000 01110001 01101000 01111001 01110010 01101111
01101000 01110001 01110111 00101100 00100000 01101001 01110101 01101000 01101000 00101101 01110110 01110011 01101100 01110101 01101100 01110111 01101000
01100111 00100000 01100100 01110001 01100111 00100000 01110010 01110011 01101000 01110001 00101101 01110000 01101100 01110001 01100111 01101000 01100111
00101100 00100000 01111010 01101011 01101100 01100110 01101011 00100000 01101111 01101000 01100111 00100000 01110111 01110010 00100000 01110000 01100100
01110001 01100010 00100000 01110010 01101001 00100000 01110111 01101011 01101000 00100000 01110000 01110010 01110110 01110111 00100000 01101100 01110001
01110111 01101000 01110001 01110110 01101000 00100000 01100100 01110001 01100111 00101111 01110010 01110101 00100000 01100100 01100101 01110110 01111000
01110101 01100111 00100000 01100111 01101100 01110110 01100110 01111000 01110110 01110110 01101100 01110010 01110001 01110110 00100000 01001100 00100000
01101000 01111001 01101000 01110101 00100000 01101011 01100100 01100111 00101100 00100000 01101010 01110010 01101100 01110001 01101010 00100000 01101001
01110101 01110010 01110000 00100000 00100010 01001011 01110010 01111010 00100000 01100110 01100100 01110001 00100000 01111010 01101000 00100000 01110101
01100100 01101100 01110110 01101000 00100000 01100100 01110001 01100111 00100000 01100110 01110010 01110001 01110111 01110101 01110010 01101111 00100000
01100101 01111000 01110111 01110111 01101000 01110101 01101001 01101111 01101100 01101000 01110110 00100000 01101100 01110001 00100000 01110010 01110101
01100111 01101000 01110101 00100000 01110111 01110010 00100000 01110101 01101000 01110000 01110010 01110111 01101000 01101111 01100010 00100000 01100110
01110010 01110001 01110111 01110101 01110010 01101111 00100000 01110111 01101011 01101000 00100000 01101001 01110010 01110101 01110000 01100100 01110111
01101100 01110010 01110001 00100000 01110010 01101001 00100000 01110111 01110010 01110101 01110001 01100100 01100111 01110010 01101000 01110110 00100010
00100000 01110111 01110010 00100000 00100010 01001100 01110110 00100000 01101001 01110101 01101000 01101000 00100000 01111010 01101100 01101111 01101111
00100000 01110101 01101000 01100100 01101111 01101111 01100010 00100000 01101001 01110101 01101000 01101000 00101100 00100000 01100100 00100000 01100110
01110010 01110001 01110110 01101000 01110100 01111000 01101000 01110001 01100110 01101000 00100000 01110010 01101001 00100000 01100110 01110010 01110000
01110011 01101111 01101100 01100110 01100100 01110111 01101000 01100111 00100000 01100101 01111000 01110111 00100000 01100111 01101000 01110111 01101000
01110101 01110000 01101100 01110001 01101100 01110110 01110111 01101100 01100110 00100000 01110011 01101011 01100010 01110110 01101100 01110010 01101111
01110010 01101010 01101100 01100110 01100100 01101111 00100000 01100100 01110001 01100111 00100000 01110011 01110110 01100010 01100110 01101011 01110010
01101111 01110010 01101010 01101100 01100110 01100100 01101111 00100000 01110011 01110101 01110010 01100110 01101000 01110110 01110110 01101000 01110110
00101100 00100000 01110010 01110101 00100000 01100100 00100000 01110100 01111000 01100100 01110001 01110111 01111000 01110000 00100000 01101100 01101111
01101111 01111000 01110110 01101100 01110010 01110001 00111111 00100010 00100000 01100101 01100010 00100000 01111010 01100100 01100010 00100000 01110010
01101001 00100000 00100010 01001011 01110010 01111010 00100000 01110111 01110010 00100000 01110000 01100100 01101110 01101000 00100000 01101111 01100100
01111010 01110001 01110000 01110010 01111010 01101000 01110101 00100000 01110011 01110010 01111010 01101000 01110101 01101000 01100111 00100000 01100101
01100010 00100000 01101011 01110010 01111000 01110110 01101000 00101101 01110000 01100100 01100111 01101000 00100000 01110000 01101100 01110001 01101100
01100100 01110111 01111000 01110101 01101100 01100011 01101000 01100111 00100000 01110001 01111000 01100110 01101111 01101000 01100100 01110101 00100000
01110101 01101000 01100100 01100110 01110111 01110010 01110101 00111111 00100010 00101110 00001010 00001010 01010110 01010000 01000100 01000110 00100000
01101100 01110110 00100000 01100100 00100000 01110111 01101000 01100100 01110000 00100000 01110111 01101011 01100100 01110111 00100000 01100100 01110001
01100010 01110010 01110001 01101000 00100000 01100110 01110010 01111000 01101111 01100111 00100000 01111010 01101100 01110110 01101011 00100000 01101001

01110010 01110101 00100000 01100100 01110001 01100111 00100000 01110000 01110010 01110101 01101000 00101110 00100000 01001100 01110111 00100000 01101100
01110110 00100000 01101111 01101100 01111001 01101000 01101111 01100010 00101100 00100000 01101001 01111000 01110001 00101100 00100000 01101100 01110001
01110111 01101000 01101111 01101111 01101000 01100110 01110111 01111000 01100100 01101111 01101111 01100010 00100000 01110110 01110111 01101100 01110000
01111000 01101111 01100100 01110111 01101100 01110001 01101010 00100000 01100100 01110001 01100111 00100000 01101001 01111000 01101111 01101001 01101100
01101111 01101111 01101100 01110001 01101010 00101110 00100000 01001001 01111000 01110101 01110111 01101011 01101000 01110101 01110000 01110010 01110101
01101000 00101100 00100000 01010110 01010000 01000100 01000110 00100000 01101100 01110110 00100000 01100110 01110010 01110010 01110011 01101000 01110101
01100100 01110111 01101100 01111001 01101000 00101100 00100000 01110110 01101000 01101111 01101001 00101101 01100100 01100111 01100100 01110011 01110111
01101100 01111001 01101000 00100000 01100100 01110001 01100111 00100000 01100100 01110001 01110111 01101100 01100110 01101100 01110011 01100100 01110111
01101000 01110110 00100000 01010001 01110010 01110001 00100000 01000110 01110010 01110010 01110011 01101000 01110101 01100100 01110111 01101100 01111001
01101000 00100000 01010110 01101100 01110111 01111000 01100100 01110111 01101100 01110010 01110001 01110110 00101110 00100000 01010111 01101011 01100100
01110001 01101110 01110110 00100000 01110111 01110010 00100000 01010110 01010000 01000100 01000110 00101100 00100000 01001100 00100000 01101111 01101000
01100100 01110101 01110001 01101000 01100111 00100000 01100100 00100000 01101111 01110010 01110111 00100000 01100100 01100101 01110010 01111000 01110111
00100000 01110000 01100010 01110110 01101000 01101111 01101001 00100000 01100100 01110001 01100111 00100000 01110010 01110111 01101011 01101000 01110101
01110110 00101110 00100000 01001100 00100000 01101111 01101000 01100100 01110101 01110001 01101000 01100111 00100000 01100100 01101111 01110110 01110010
00100000 01110111 01110010 00100000 01100101 01101000 00100000 01110000 01110010 01110101 01101000 00100000 01110110 01101000 01101111 01101001 00101101
01100110 01110010 01110001 01101001 01101100 01100111 01101000 01110001 01110111 00100000 01100100 01110001 01100111 00100000 01100110 01110010 01110001
01110110 01101100 01100111 01101000 01110101 00100000 01101000 01111001 01101000 01110101 01100010 00100000 01110011 01110010 01101100 01110001 01110111
00100000 01110010 01101001 00100000 01111001 01101100 01101000 01111010 00101110 00100000 01000100 01110110 00100000 01001100 00100000 01110111 01100100
01101110 01101000 00100000 01100100 00100000 01101111 01110010 01110111 00100000 01110010 01101001 00100000 01100010 01110010 01111000 00100000 01111010
01101100 01110111 01101011 00100000 01110000 01101000 00101100 00100000 01001100 00100000 01101011 01110010 01110011 01101000 00100000 01110111 01101011
01100100 01110111 00100000 01100010 01110010 01111000 00100000 01101110 01101000 01101000 01110011 00100000 01110110 01110010 01110000 01101000 00100000
01110010 01101001 00100000 01110000 01101000 00100000 01111010 01101100 01110111 01101011 00100000 01100010 01110010 01111000 00101110 00001010 00001010
01001100 01110001 00100000 01100100 01100111 01100111 01101100 01110111 01101100 01110010 01110001 00100000 01110111 01110010 00100000 01110111 01101011
01101000 00100000 01010110 01010000 01000100 01000110 01101110 01101000 01110101 01110110 00101100 00100000 01001100 00100000 01110000 01101000 01110111
00100000 01110001 01101000 01111010 00100000 01101001 01110101 01101100 01101000 01110001 01100111 01110110 00100000 01100100 01110001 01100111 00100000
01100101 01110101 01110010 01110111 01101011 01101000 01110101 01110110 00100000 01101100 01110001 00100000 01010111 01110010 01111000 01101111 01110010
01111000 01110110 01101000 00100000 01100100 01110001 01100111 00100000 01010111 01110010 01101110 01100010 01110010 00100000 01111010 01101011 01110010
00100000 01100110 01110010 01110001 01110111 01110101 01101100 01100101 01111000 01110111 01101000 01100111 00100000 01110111 01110010 00100000 01110111
01101011 01101100 01110110 00100000 01111010 01110010 01110101 01101110 00100000 01101100 01110001 00100000 01111001 01100100 01110101 01101100 01110010
01111000 01110110 00100000 01111010 01100100 01100010 01110110 00101110 00100000 01001000 01110110 01110011 01101000 01100110 01101100 01100100 01101111
01101111 01100010 00100000 01000010 01100100 01110110 01110110 01101100 01110001 01101000 00100000 01010000 01110010 01110111 01101100 01101000 00101100
00100000 01110000 01100010 00100000 01101001 01101000 01101111 01101111 01110010 01111010 00100000 01010011 01101011 00101110 01000111 00101110 00100000
01110110 01110111 01111000 01100111 01101000 01110001 01110111 00100000 01101100 01110001 00100000 01110001 01101000 01010010 01000110 01100100 01110000
01110011 01111000 01110110 00100000 01111010 01101011 01110010 01110110 01101000 00100000 01100110 01110010 01101111 01101111 01100100 01100101 01110010
01110101 01100100 01110111 01101100 01110010 01110001 00100000 01111010 01100100 01110110 00100000 01101000 01110001 01101101 01110010 01100010 01100100
01100101 01101111 01101000 00100000 01100100 01110001 01100111 00100000 01101000 01110000 01110011 01101011 01100100 01110110 01101100 01100011 01101000
01100111 00100000 01110111 01101011 01101100 01110110 00100000 01111010 01110010 01110101 01101110 00101110 00001010 00001010 01001111 01100100 01110110
01110111 00100000 01100101 01111000 01110111 00100000 01100100 01100101 01110110 01110010 01101111 01111000 01110111 01101000 01101111 01100010 00100000
01110001 01110010 01110111 00100000 01101111 01101000 01100100 01110110 01110111 00101100 00100000 01001101 01101000 01100100 01110001 00101101 01010011
01101100 01101000 01110101 01110101 01101000 00101100 00100000 01100100 01110110 00100000 01100100 00100000 01110111 01101000 01100100 01100110 01101011
01101000 01110101 00100000 01100010 01110010 01111000 00100000 01110111 01100100 01111000 01101010 01101011 01110111 00100000 01110000 01101000 00100000
01000100 01010000 01000100 01010110 00100000 01100100 01110101 01101000 00100000 01101001 01111000 01110001 00100000 01100100 01110001 01100111 00100000
01100100 01110000 01100100 01100011 01101100 01110001 01101010 00101110 00100000 01000100 01110110 00100000 01100100 00100000 01110110 01111000 01110011
01101000 01110101 01111001 01101100 01110110 01110010 01110101 00100000 01100010 01110010 01111000 00100000 01110110 01111000 01110011 01110011 01110010

01110101 01110111 01101000 01100111 00100000 01110000 01101000 00100000 01100100 01101111 01101111 00100000 01110111 01101011 01101000 00100000 01111010
01100100 01100010 00100000 01100100 01110001 01100111 00100000 01110000 01110010 01110101 01101000 00101100 00100000 01100010 01110010 01111000 00100000
01101011 01100100 01100111 00100000 01100100 01101111 01111010 01100100 01100010 01110110 00100000 01110111 01101011 01101000 00100000 01110101 01101100
01101010 01101011 01110111 00100000 01111010 01110010 01110101 01100111 01110110 00100000 01110111 01110010 00100000 01100110 01101011 01100100 01110001
01110001 01101000 01101111 00100000 01110000 01100010 00100000 00100010 01101000 01110001 01101000 01110101 01101010 01100010 00100010 00100000 01100100
01110001 01100111 00100000 01110101 01101000 01100100 01110110 01110110 01111000 01110101 01101000 00100000 01110000 01101000 00101110 00100000 01000100
01110110 00100000 01100100 00100000 01110000 01101000 01110001 01110111 01110010 01110101 00101100 00100000 01100010 01110010 01111000 00100000 01110110
01101011 01110010 01111010 01101000 01100111 00100000 01110000 01101000 00100000 01111010 01101011 01100100 01110111 00100000 01101110 01101100 01110001
01100111 00100000 01110010 01101001 00100000 01110101 01101000 01110110 01101000 01100100 01110101 01100110 01101011 01101000 01110101 00100000 01001100
00100000 01100100 01101100 01110000 00100000 01110111 01110010 00100000 01100101 01101000 00101110 00100000 01000100 01110001 01100111 00100000 01101001
01110010 01110101 00100000 01110111 01101011 01101000 00100000 01101001 01111000 01110111 01111000 01110101 01101000 00100000 01100100 00100000 01101001
01101000 01101111 01101111 01110010 01111010 00100000 01100100 01110001 01100111 00100000 01100100 00100000 01101001 01110101 01101100 01101000 01110001
01100111 00101110 00001010 00001010 01001001 01101100 01110001 01100100 01101111 01101111 01100010 00101100 00100000 01001100 00100000 01100111 01101000
01100111 01101100 01100110 01100100 01110111 01101000 00100000 01110111 01101011 01101100 01110110 00100000 01110111 01101011 01101000 01110110 01101100
01110110 00100000 01110111 01110010 00100000 01110000 01100010 00100000 01101001 01100100 01110000 01101100 01101111 01100010 00100000 01100100 01110001
01100111 00100000 01100101 01110101 01110010 01110111 01101011 01101000 01110101 01110110 00100000 01111010 01101011 01110010 01110110 01101000 00100000
01101111 01110010 01111001 01101000 00101100 00100000 01110011 01110101 01100100 01100010 01101000 01110101 01110110 00101100 00100000 01100100 01110001
01100111 00100000 01110110 01110011 01101100 01110101 01101100 01110111 01110110 00100000 01100100 01100110 01100110 01110010 01110000 01110011 01100100
01110001 01101100 01101000 01100111 00100000 01110000 01101000 00100000 01110111 01101011 01101000 00100000 01111010 01101011 01110010 01101111 01101000
00100000 01110111 01101100 01110000 01101000 00100000 01110010 01111001 01101000 01110101 01110110 01101000 01100100 01110110 00101110

Many Thanks...

Hadi.

Contents

General Introduction

SINCE its beginning, humankind seeks to understand its environment and the world wherein it evolves to satisfy its curiosity and increase its well-being. This will to discover and understand phenomena led to science and the scientific method. The scientific method can be summarized as: first make an hypothesis, then establish an experiment and observe its result by repeating the same experiment with different parameters to confirm or deny the hypothesis. Changing the experiment parameters allows to study cause and effect relations and exploit them to come up with new tools, products, and services in every field to ease life and improve resources consumption.

Finding relations between factors, involved in physical, chemical, biological, psychological, economical, social mechanisms and others, begins with looking for correlations between them, then conduct new experiments to explain these correlations and finally concludes which ones are true relations, spurious relations and correlations between child-factors related to the same known or unknown parent-factor. Hence the need to record every aspect of the experiment and each observation and study them afterward. These records, called *data*, come in various shapes and on different supports that evolve over time. This is a thorough and difficult process, during which many mistakes can easily happen because of the human limitations. In order to facilitate the data management (gathering, storage, and analysis) and to reduce the number of potential errors, the use of machines, more specifically computers, has been growing and popularized, considering their increasing reliability and efficiency.

Nowadays, computers and computer science are present in every and each science experiment to manage the data. As a result, the volume of data has been increasing and a new field of numerical science, the *Data Science*, emerged for the sake of automatic data gathering, storage, and analysis. Data Science is a multidisciplinary field that relies mainly on mathematics, statistics, and computer science to provide tools that help their user to understand the data and extract its own knowledge about them. However, despite data science continuous progression, the gap between the capabilities of its tools and their efficiency is widening as a consequence of the data flood: the exponential growth of the data volume, speed, heterogeneity due to the ever improving and spreading technologies that generate, record and carry out new data. Hence the urge to design new tools able to manage the data flood in new ways.

Contribution of the Thesis

The purpose of this thesis is to provide an original analytics framework to handle the data flood as a complex system, with strong dynamics, quick expansion, and frequent anomalies, rather than focus on improving the hardware and software architecture of the current analytics tools. To do so, I rely on an existing bio-inspired artificial collective intelligence, the *adaptive multi-agent system (AMAS)*, which has been shown as particularly relevant for modeling, simulating and solving problems in highly dynamic complex systems.

This new analytics framework, called *Adaptive Multi-Agent System for dynamic Big Data analytics (AMAS4BigData)*, has been used to design a new analytics tool, *Dynamic data Relation Extraction using Adaptive Multi-agent system (DREAM)*, which processes in real time huge amounts of data produced by several and different sources, like sensors data from *Internet of Things (IoT)*, and find relations between them with an adaptation to the data changes on the fly and respect of the data providers privacy and anonymity.

The dynamic data relation extraction process studies the dynamics correlations between data, which means finding the data that evolve and behave in a similar way even if they are temporally shifted, thanks to a new data similarity metric developed to deal with the data dynamics in real time.

Since this thesis is funded by the Occitanie region, whose one major goal is to develop smart cities, DREAM is applied on *neOCampus* sensors data in order to contribute to that effort. The *neOCampus* operation, lunched by the University of Toulouse III - Paul Sabatier, provides a testing ground for multi-disciplinary tools in the scope of building self-sustainable smart cities. Therefore, DREAM is used to find hidden relations between the campus sensors and detect anomalies, in order to understand and improve human activities while decreasing the campus energy consumption.

Manuscript Organisation

This manuscript is composed of three main parts. The first one presents the context and the field application of this thesis, the Data Science and the motivations behind this work. Part 1 is divided in four chapters:

- Chapter 1: introduces the data science field, by presenting its history and origins, its definitions, main concepts, and its uses.
- Chapter 2: summarizes in one global overview the process of knowledge extraction from data (data analytics). Then it presents the step of the process and gives it limitations.
- Chapter 3: explains the theoretical and practical evolution of the data analytics pipeline to the big data analytics pipeline, by describing thoroughly the concept of data science on which I focus my work, the big data analytics which the set of techniques and processes that allow the knowledge extraction from big data.

- Chapter 4: in this chapter, I give the current data science challenges and I define new ones to reflect what data science needs to handle the data flood dynamically.

Part 2 gives the theoretical and technical foundations and the formal description of *AMAS4BigData*, the new analytics framework that combines the *Adaptive Multi-Agent System (AMAS)* technology with a new dynamic Big Data similarity metric, the *Dynamics Correlation*. This part is detailed through the following chapters:

- Chapter 5: first highlights the need of a new big data analytics paradigm able to handle the complex nature of the big data. Then describes this new paradigm of big data analytics with the help of the Adaptive Multi-Agent System theory.
- Chapter 6: presents the Adaptive Multi-Agent System (AMAS) theory, an artificial collective intelligence theory, whose compounds have been tailored to model my new analytics framework, which is thoroughly described here.
- Chapter 7: describes the *Dynamic data Relation Extraction using Adaptive Multi-agent system (DREAM)* analytics tool, the main application implemented using the AMAS4BigData framework and a new data similarity metric, the *Dynamics Correlation*.

The last part, Part 3, issues the experimental aspect of this thesis through two chapters.

- Chapter 8: describes the experimental protocol used to evaluate the framework, thanks to the presentation of the data sets used to test the AMAS4BigData framework, via DREAM, and the definition of criteria for evaluating its performances.
- Chapter 9: embodies my study of the new analytics framework performances, thanks the disscussion of the results of several experiments conducted on various data sets with different parameters. Finally, it displays the strengths and weaknesses of AMAS4BigData in order to know when its use is best suited.

Finally, I conclude this manuscript with a synthesis of the work achieved in the thesis and an enumeration of some interesting further works.

AMAS4BigData
Adaptive Multi-Agent Systems
for Dynamic Big Data Analytics

Data Science

Abstract

UNDERSTANDING data and extract for them relevant knowldge is the main purpose of Data Science. Achieving this purpose raises several challenges and issues, besides the usual matter of time and space use efficiency, especially when dealing with big data. In order to find meaning and relevant information drowned in the data flood, while overcoming big data challenges, one should rely on analytics tool that crunch these data and present them in a readable manner to the user.

Part I presents the *Data Science* field through three chapters. The first one looks its origin up across history, gives a set of definitions and keywords to get rid of any ambiguity, due to their misuse, and help the reader to distinguish between close concepts and get a better grasp of data science goals, listed at the end of the chapter.

Chapter 2 dives into the theoretical and technical aspects and concepts of the data science. First, this chapter furnishes an overview of the data science concepts, then focuses on the data analysis workflow, called *"Big Data Analytics"*. It goes through it building blocks and presents the state of art techniques and tools that design them and/or use them to realize various tasks applied in several domains.

The first part concludes by presenting the conventional data science issues, known as *"challenges"*, that have led to the current data analytics approaches. It infers new challenges as response to the modern expectations of the data science and to cope with exponential growth of the digital world.

1 Introduction to Data Science

Nowadays, we are living in a *cyberphysical* world [1]: the cyber, or "digital", world is a projection of the physical "real" world. This projection is a collection of digitized[1] information, the *"data"*, created by all the physical and virtual devices[2] that record the state and activity of the physical world.

These two worlds are overlapping and deeply intertwined. Put it simply, they interact and affect each other. In addition, they operate on different spatial and temporal scales, meaning that the cyber world requires less time and space to represent an entity, be it static or dynamic. So, through the study of the cyber world and its data we can comprehend, in a faster and easier way, the real world. This is known as *Data Science*.

Data science is a composite of a number of pre-existing disciplines. It is a young academic discipline and so is the associated profession of data scientist. The term, with its current meaning, was essentially coined at the beginning of the 21st Century [2]. About a year later, the *International Council for Science: Committee on Data for Science and Technology* [3] started publishing the *CODATA Data Science Journal*[4] beginning April 2002. Shortly thereafter, in January 2003, Columbia University began publishing The *Journal of Data Science* [5].

Data science popularity has exploded since 2010, pushed by the need for teams of people to analyze the big data that corporations and governments are collecting. The Google search engine is a classic example of the power of data science.

[1] Converting analogous physical measurements to computer-readable format, in which the information is organised into bits, basic units of information used in computing and digital communications.

[2] Ambient sensors, cameras, Internet logs, transactions logs, etc.

1.1 Definitions & Keywords

First of all, in order to get a better grasp of this thesis context, which is data science generally and big data analytics specifically, and avoid confusion among several similar terms, I present in this section the definition of these keywords from the literature. I also redefine some for the sake of better relevance and less disarray.

- **Data**: it is the digital projection of a broad information. The *Oxford English Dictionary (OED)* define *"data"*, in non-technical contexts, as information in digital[1] form, typically obtained by scientific work and used for reference, analysis, or calculation. In computing, *"data"* refers to quantities, characters, or symbols on which operations are performed by a computer, considered collectively [6]. Even if *"data"* is the plural of *"datum"*, *"data"* is widely used as an uncountable noun.
- **Big Data**: today *"Big Data"* still has two meanings. The first one implies the huge and fast growing amount of data [7][8]. The second one, is the study field of storing and managing easy access to the data, and sometimes it is mistaken with the data mining. Therefore in my opinion, the second definition is less accurate and less relevant, especially since we have other dedicated terms (data storing, data warehousing, data base managing).
- **Big Data Analytics** [9]: analytical processes or workflows that associate advanced and unique data storage, management, mining, and visualization technologies and techniques in order to analyze big data.
- **Business Intelligence (BI)**: Negash Solomon gave, in his article *"Business Intelligence"* [10], an extensive definition of this concept. I summarize BI as big data analytics, focused on data gathering, integration and visualization, to present complex and competitive information to planners and decision makers, for the sake of improving the timeliness and quality of inputs to decision processes.
- **Data Mining**: is the practice of discovering patterns[3] in large data sets involving methods at the intersection of machine learning, statistics, and database systems [11]. As part of data science and main step in the *KDD* process (which is defined bellow), *data mining* aims to *"crunch"* and explore data in order to find relevant information, so that the users (data miners) produce their own knowledge about the data.
- **Knowledge Discovery in Databases (KDD)** [11]: the overall process of discovering useful knowledge from data, in which data mining is only a particular. Moreover, the additional steps in the *KDD* process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data.

  The generalization and extension of *KDD* to take into account the newly defined big data (not only the non-volatile and structured data present in databases) coming from several heterogeneous sources, is the root of the *Big Data Analytics*.

[3] Regular and intelligible forms or sequences discernible in the way in which something happens or is done.

▷ **Data Science**: many definitions are available in the literature, some more precise than others, with no consensus. In this thesis I define *data science*, based on the definition of its related terms and concepts, as a discipline that incorporates varying degrees of data engineering, scientific method, mathematics, statistics, artificial intelligence, advanced computing, visualization, hacker mindset[4], and domain expertise (as illustrated in fig.1.1) to design, implement[5], improve, adapt and extend big data analytics, partially or entirely, while overcoming the associated challenges (cf. chapter 4).

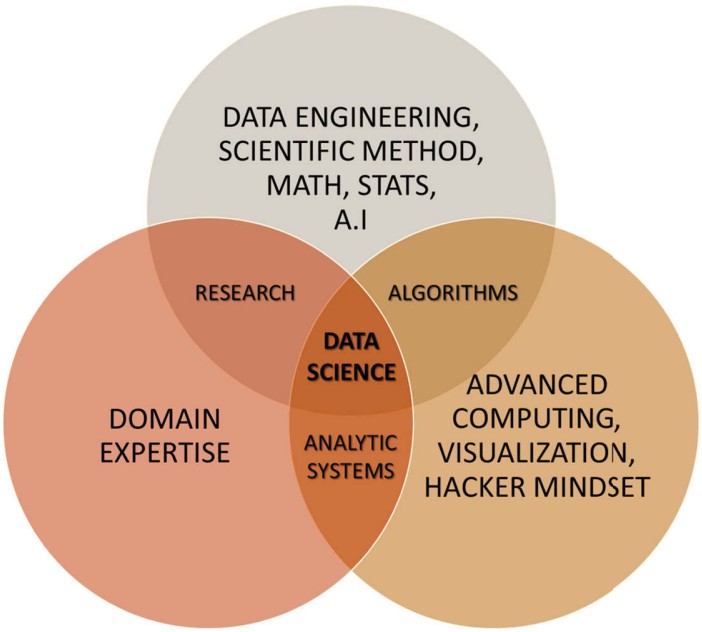

Figure 1.1 — Concepts related to data science.

▷ **Data Scientist**: a practitioner of data science. Data scientists solve complex data analysis problems. Simberloff, D., et al defined data scientists as the information and computer scientists, database and software engineers and programmers, disciplinary experts, curators and expert annotators, librarians, archivists, and others, who are crucial to the successful management of a digital data collection [12].

▷ **Data Analyst**: a user of data science tools to analyze data.

1.2 Origin of Data Science

Making sense of data has a long history and has been discussed by scientists, statisticians, librarians, computer scientists and others for years. The following time-line, which relies on *"A very short history of data science"* [13] and summarized in fig.1.2, traces the origin of the term *"Data Science"*, its use, attempts to define it, and related terms.

[4] Thinking *"outside the box"* and joyfully taking up intellectual challenges to achieve novel and clever outcome.

[5] Put into effect according to or by means of a definite plan or procedure.

- **1962**: John Tukey observed that a shift in the world of statistics has started. As a response to this shift, he came up with a first definition of data science through his experience and his view of the data science challenges [14]. He also referred to the merging of statistics and computers, at a time when statistical results were presented in hours, rather than the days or weeks it would take if done manually. Later in 1974, Peter Naur authored the book *"Concise Survey of Computer"* [15], a survey of contemporary data processing methods that are used in a wide range of applications. Naur had used the term "data science" repeatedly and defined it as: "The science of dealing with data, once they have been established, while the relation of the data to what they represent is delegated to other fields and sciences."

- **1977**: *The International Association for Statistical Computing (IASC)* [16] was formed with the mission *"to link traditional statistical methodology, modern computer technology, and the knowledge of domain experts in order to convert data into information and knowledge"*. The same year, Tukey published *"Exploratory Data Analysis"* [17], a book wherein he argues that more emphasis and interest must be placed on using data to suggest which hypotheses have to be tested, adding that exploratory data analysis and confirmatory data analysis can and should proceed side by side.

- **1989**: Gregory Piatetsky-Shapiro organises and chairs the first *Knowledge Discovery in Databases (KDD)* workshop [18], which would become in 1995 into *the annual ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

- **1994**: Jonathan Berry alluded to the *data flood* when he wrote a cover story on *"Database Marketing"* [19], in which he revealed ominous news that companies had started gathering large amounts of personal data, with plans to start precisely calibrated new marketing campaigns to increase the likelihood of customer buying a product. The flood of data was, at best, confusing to company managers, who were trying to decide what to do with so much disconnected information.

- **1996**: for the first time, the term *"data science"* was included in the title of a conference (*"Data science, classification, and related methods"*) [20] of the *International Federation of Classification Societies (IFCS)*. The IFCS have variously used the terms data analysis, data mining, and data science in their publications.

  Later that year, Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth pointed out that a lot of terms were used, indiscriminately, to express the same idea of *"finding useful patterns in data"*. For this reason, they defined the *KDD (Knowledge Discovery in Databases)*, in their article *"From Data Mining to Knowledge Discovery in Databases"* [11] (defined in section 1.1). Furthermore, they stressed that blind application of data mining methods can be a dangerous activity, easily leading to the discovery of meaningless and invalid patterns.

- **2002**: CODATA [3] began publishing the Data Science Journal [4], wherein a publication focused on issues such as the description of data systems, their publication on the internet, applications and *legal issues*.

▷ **2005**: the term *"data scientist"* appeared [12]. Starting 2008 it became a buzz word and describe as the sexy job of the next years [21][22][23].

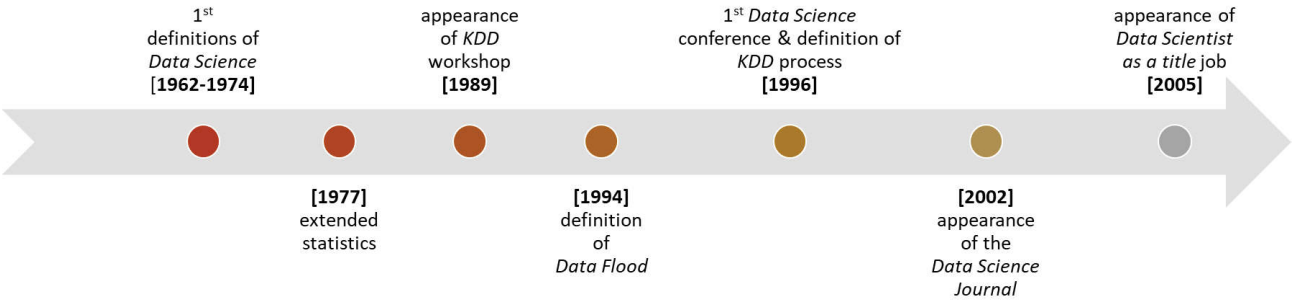

Figure 1.2 — Time-line of Data Science.

At the end of the 20^{th} Century, data science emerged as a logical response to the need to extend *Information Technology (IT)*[6], in order to take advantage of the big data. Since then, data science is still growing, evolving, and incorporating new technologies coming from various sciences, which exploit data science to their benefit. Thus, data science is interdisciplinary and has a symbiotic relationship with the other sciences, meaning they sustain each other and help each other to grow.

1.3 Goals and Uses of Data Science

The explosive growth of available data, as a result of the increasing computerization of our societies and the fast development and production of powerful data generation, collection and storage tools, impacts in different degrees businesses, societies, sciences and engineering, medicine, and almost every other aspect of daily life.

Objectively speaking, data science aims to provide powerful and versatile tools to automatically uncover valuable information from the tremendous amounts of data, produced continuously, and to transform these data into useful knowledge. However, realistically, this knowledge can be used for "good" deeds or "bad" deeds.

In this section, I present non-exhaustive lists of positive and negative data science uses, with some examples, and the purposes behind them.

1.3.1 Positive Aspects

The original objective of data science is to ultimately improve human life, by means of the improvement of sciences, governance, business, transportation, health care and other areas as the ones listed hereafter.

▷ **Scientific research**: today almost every science relies on data science [24][25], especially the ones that produce tremendous data like bioinformatics [26], computational biology

[6] The study or use of systems (especially computers and telecommunications.) for storing, retrieving, and sending information.

[27], social computing [28] and physics. For example the *Large Hadron Collider (LHC)*[7] can generate 60 terabytes (TB[8]) of data per day [29] and using data science helps to uncover patterns in the generated data which can give the physicists an unprecedented understanding of the nature of the universe.

- **Health care**: large amount of data are generated from physician notes, lab reports, X-Ray reports, case history, national health register data and others. Health care organisations are progressively relying on data science technologies to capture all these patients data to reduce health care expenditure. For instance the use of data science could save over EUR 100 billions for the E.U. health care [30], while providing better:

  - patient-centred services, like faster relief, detecting diseases at the earlier stages, minimizing drug doses to avoid side effect and thereby reducing readmission rates and health care costs consequently [31].
  - spreading diseases detection, based on the live analysis of the social logs and search engine queries related to a disease in a particular geo-location. For example, in 2009 during the flu pandemic, Google obtained timely information by analyzing big data, which even provided more valuable information than that provided by disease prevention centers. Google found 45 search entry groups that were closely relevant to the outbreak of influenza and incorporated them in specific mathematics models to forecast the spreading of influenza and even to predict places where influenza spread from [32].
  - health management, patient engagement and treatment methods [33]. This is realized, on one hand by giving patients the most timely and appropriate treatment available through the coordinated effort of the most skilled caregivers, according to their recorded ability to achieve the best outcomes rather than their job titles, who avoid sub-optimal treatment strategies, thanks to sharing and logically centralizing patients information. On the other hand, encouraging patients to play an active role in their own health by making the right choices about diet, exercise, preventive care, and other lifestyle factors.

  This shift in the US health care sector results in the development of many innovative data applications that move beyond retroactive reporting to interventions and predictive capabilities (fig.1.3).

- **Governance, Smart Urbanism and Smart Cities**: governments also generate lot of data every day. Data science helps governments to make valuable decisions and build smart cities [34] by providing faster, reliable and value-added services to the citizens.

  - *Society Administration* [31]: collecting and analyzing citizens status and activity data improve the calculation of seniors pensions and provide them without any delay. Data science also provides quality education, by getting detailed report of children who are in the age to be admitted to the school and which one is the best

[7] CERN's famous particle accelerator.
[8] 1 TB = 10^{12} bytes = 1000 gigabytes.

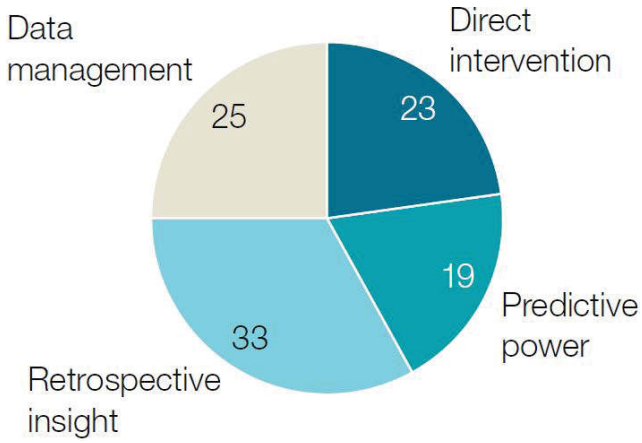

Figure 1.3 — US health care data applications from 100 top innovators by type of data/analytics capability, 2010-12 [33].

suited for them, in accordance with for example their living area, their parents work area and schools enrolling capacity.

Moreover, the analysis of student profiles, graduating each year, can reduce the unemployment rate by predicting the job needs based on the literacy rate and arrange for special trainings in order to build young professionals.

– *Safety and Security*: a lot of city data, came from surveillance devices, tracking devices and others, allow to screen the various activities taking place in the city and can be used to detect abnormal behaviors, thence trigger alerts when the situation is hazardous.

  For instance, automatic number plate recognition systems that use digital cameras to scan license plates and trace the matched vehicles as they cross the city. Similarly, the police might screen a suite of cameras and live incident logs in order to efficiently and swiftly direct appropriate resources to particular locations [35].

  City data can also be very helpful to handle crises and emergencies, like understanding the situations happening in disaster areas in order to effectively provide the best communication services and optimally deploy the limited resources based on the analysis results [36].

– *Transportation and Mobility*: perhaps the most critical feature of a city, is its ability to control traffic in peak times based on the live streaming data about vehicles and supply a smooth public transportation network to support a comfortable mobility for the citizens.

  This goes by, for example, transponders that can be used to monitor throughput at toll-booths, measuring vehicle flow along a road or the number of empty spaces in a car park, and track the progress of buses and trains along a route. Moreover, these transponders alongside with data of cameras network can be analyzed in real-time to adjust traffic light sequences and speed limits and to automatically administer penalties for traffic violations [37].

  Furthermore, the citizens can be active parts in the mobility enhancement thanks to the crowdsourcing of data, wherein users generate data and then contribute

them to a common system, such as the generation of GPS-traces uploaded into *OpenStreetMap*[9] to create collaboratively a common, open map of the city [38]. Sometimes, such collaborative solutions can best the conventional ones, since the local users have unique and expert knowledge of their areas.

– *Waste Management and Energy Saving*: another advantage of data science applied on smart cities is to increase their self-sustainability via efficient waste management, using the automated monitoring of public service provision, such as RFID[10] chips attached to rubbish bins detecting whether they have been collected [35]; data sharing between truck drivers on real time to perform waste collection, dynamic route optimization and detecting ineffective collection in inaccessible areas within the Smart City. Surveillance cameras are incorporated for capturing the problematic areas and provide evidence to the authorities [39].

  Energy saving is currently the most essential feature of smart cities, due to financial and environmental benefits. Many tools and systems are developed to achieve this goal, such as automatic meter reading that communicates utility usage without the need for manual reading and can do so on a continuous basis [37]. This kind of systems are very useful to reduce citizen energy consumption using *eco-feedbacks*[11] [40].

  One additional important set of systems is the *Smart Grid* [41] and *Zero Energy Buildings* [42], which promises to bring green revolution by integrating renewable resources into the energy mainstream and reducing the energy demand of the buildings, by means of ambient intelligence that autonomously manages heating, cooling, venting, lighting and other energy consuming equipments and services based on the human activity patterns and energy consumption behaviors uncovered from heterogeneous sensors data and automatic meter readings.

Recently there have been attempts to draw all of these systems and analytics tools into one huge single hub, which facilitate the city monitoring and management from one central place.

For instance, the *Centro De Operacoes Prefeitura Do Rio* in Rio de Janeiro, Brazil, uses an IBM product that can be applied to any city, called *"IBM Intelligent Operations Center" (IOC)* [43]. This IOC combines city wide instrumented systems that draws together data streams from thirty agencies, including traffic and public transport, municipal and utility services, emergency services, weather feeds, and information sent in by employees and the public via phone, internet and radio, into a single data analytics centre.

▷ **Business**: the financial profit was and still is the first driving force for the data science development. Data science is particularly helpful for building new capabilities for exploiting data and facilitate decision making [44].

[9] A collaborative project to create a free editable map of the world.

[10] Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects.

[11] Information that helps to visualize energy consumption.

There are many advantages in the business area [45], which can be obtained through harnessing big data, include increasing operational efficiency, informing strategic direction, developing better customer service, identifying and developing new products and services, identifying new customers and markets, etc (see fig.1.4 wherein the vertical axis denotes the percentage of 560 enterprises that think big data can help them).

For example, the famous retail stores Wal-Mart successfully improved the efficiency of their pricing strategies and advertising campaigns [30], by applying machine learning on their huge data warehouse, constructed with the data of every purchase record from their point-of-sale terminal, in order to better understand purchasing behaviour of their customers.

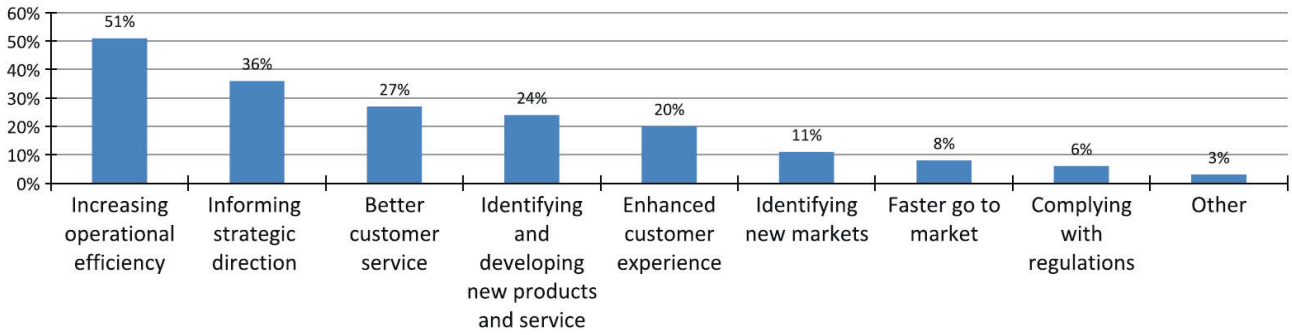

Figure 1.4 — Increasing businesses efficiency with data science [45].

1.3.2 Negative aspects

As any tool, data science is as a good and benevolent as it users. So, unfortunately, data science can be used for reprehensible purposes, deliberately or not, and lead to security leaks, social issues and injustice. Hereafter is a list of some negative uses and aspects of data science.

- *Politics*: with the current state of data science, politicians can use the data collected from social networks, web sites logs and even geo-localization devices to profile the citizens and thereby discover which one are more or less keen to vote for them and hold their ideas up. Besides the intrusive nature of such conducts, discussed later, there is a huge ethical issue of impacting elections result.

  For example in 2012 during the Obama campaign, data were first crossed from the 2008 voters database, the posts on Obama's web sites and social network profiles of the voters. The next step was to sort the voters, based on their data, into three categories, those in favor of Obama, those against and those in-between. Then, speeches were automatically generated using *Natural Language Processing* and *Opinion mining* to focus strategically on the elements that might attract those in between, even though they were weak and minor elements of the Obama's politic, while hiding or lessening its strong elements that repel the hesitant category [46].

- *Injustice*: Jeffrey Alan Johnson shows that open data[12] has the quite real potential to exacerbate rather than alleviate injustices [47]. In 2006, the city board of Memphis (Tennessee, U.S.) equipped its law enforcement department with *"Blue Crush"* an analytics system that analyzes years of crime data to generate daily probability reports about when and where certain types of crimes are most likely to happen [48]. Even if this system reduced the overall probability of crimes occurring in Memphis, it produced two side effects. The first one is the growth of the *"being too much watched over by the police"* feeling in the disadvantaged areas and therefore, intensifies the segregation feeling and peoples stress. The second side effect is a security leak, described in the next point.

- *Security leaks*: all computer systems are hackable, to some extent. So, the big data analytics tools have some security issues that can be exploited to do some harm, be it simply spying or corrupting data and their analysis results, like accessing the *"Blue Crush"*, presented previously, to know where the police will most likely patrol and thus where most likely not, or to temper the data and skew the patrols planning. In addition, Hill warns that literally hardwiring urban services to a particular device, a particular operating system, like the *Centro De Operacoes Prefeitura Do Rio* [43], is a recipe for disaster, not efficiency [49].

- *Violation of privacy*: taking full advantage of the available data and discovering more complex and hidden relations, require big data analytics tools to cross these data. This is especially true when researchers combine multiple large data sets, that are not meant to be combined. Many studies revealed how public databases can be combined to produce serious privacy violations, such as revealing an individual's Social Security number [50], the pregnancy of teenage girls [51], the political predilection, like during the 2012 Obama's campaign [46], and other aspect of people's private life.

In summary, the abundance of data, coupled with the need for powerful data analysis tools, has been described as a *data rich but information poor* situation. The fast growing, tremendous amount of data, collected and stored in large and numerous data sets, has exceeded our human ability for comprehension without powerful tools.

As a result, the data science field appeared and is still growing and evolving, in joint efforts with other fields, to provide big data analytics tools that can turn the big data into *"golden nuggets"* of knowledge and consequently provide myriad of useful and helpful products, tools and services in almost every aspects of life. Nonetheless, this new *power* is a two sided blade and if not yielded carefully can be, directly or indirectly, harmful to individuals and society.

[12] Data made available and accessible for the public.

2 Data Analytics: Knowledge Discovery from Data

The expansion of Internet and the massive use of on-line services and tools (social networks, forums, search engines, content sharing...) led us to the data age.

Therefore, to make use of those data and extract new relevant knowledge from it, the scientists from all fields (statistics, computer science, physics, biology...) had to share and combine their analytical skills and tools for the sake of designing new methods for dealing with such data more than just, in a manner of speaking, summarizing and plotting them into charts. The data analytics field, known as *"Knowledge Discovery from Data (KDD)"*, was born.

The purpose of this chapter is to present the data analytics processes, which have been defined by different *IT*[1] communities that take part in data science research, describe their *building blocks* and their limitations.

[1] Information Technology.

2.1 Definitions

Many definitions of *Data Analytics*, usually called *Knowledge Discovery from Data (KDD)*, are available. Each definition corresponds to the community that gives it. Despite of how it may sound, this multiplicity of definitions is not confusing nor contradictory. It simply displays the importance of data analytics for various communities and how each community tries to exploit, at best, its skills and specific features in order to provide an understandable and usable data analytics process.

First, lets see the definition of data analytics, according to the main communities that do data science.

2.1.1 KDD in Data Bases

G. Piatetsky-Shapiro defined first the *Knowledge Discovery from Data (KDD)* process [18], which became the standard data analytics process. The most important step of the KDD process is finding hidden patterns through data mining. These patterns are used to build *"Models"*, a more compact or a more useful representation of the raw data.

Then, Usama Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth relied on the practical point of view of Brachman and Anand [52] in order to give the following steps, which compose the KDD as an interactive and iterative process [11]:

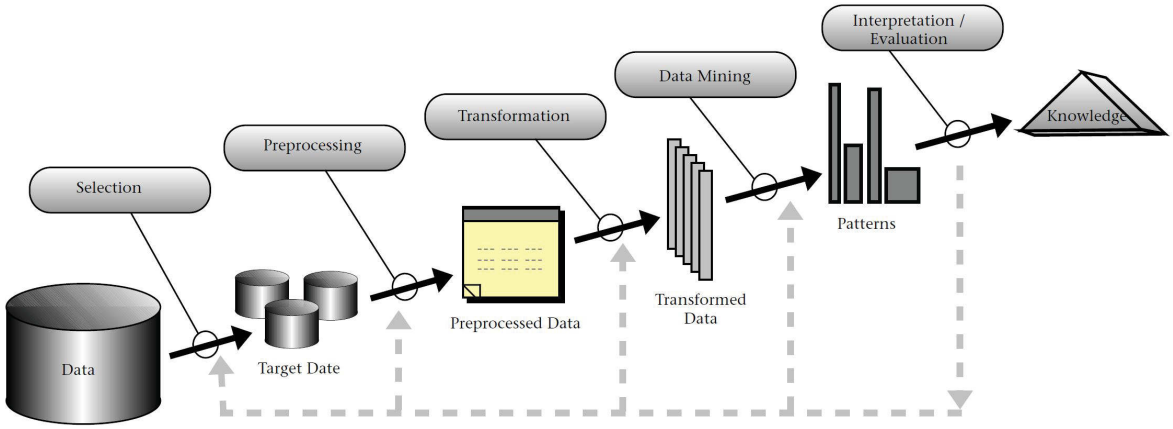

Figure 2.1 — Basic flow of knowledge Discovery from Data (KDD) [11].

1. **Selection**: retrieve data relevant to the analysis from the database on which discovery is to be performed.

2. **Pre-processing**: data cleaning and data integration, meaning the removal of noisy[2] and inconsistent[3] data combined from multiple data sources. The resulting data are usually stored in a data warehouse.

[2] Contains false, impossible or missing value.
[3] Two or more statements are inconsistent with each other if they can notall be true simultaneously.

3. **Transformation**: the step where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations. Sometimes data transformation and consolidation are performed before the data selection process. Data reduction may also be performed to obtain a smaller representation of the original data without sacrificing its integrity.

4. **Data Mining**: the most important step of the KDD process and pertains to finding hidden patterns through intelligent methods. This step includes choosing the right data mining algorithm(s) and the adequate models and parameters, that might be appropriate for searching patterns of interest, in a particular representational form or a set of such representations.

5. **Interpretation/Evaluation**: visualizing the extracted patterns to identify the truly interesting ones, that represent knowledge, based on their evaluation using relevance measures. This step can also involve acting on the discovered knowledge, which means using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties, and checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

The KDD process can involve significant iterations and can contain loops between any two steps. The basic flow of steps (besides the potential multitude of iterations and loops) is illustrated in fig.2.1.

2.1.2 KDD in Business Intelligence

The data analytics process for Business Intelligence (BI) is a derivative of the original KDD that follows the knowledge pyramid -a hierarchical representation that reflects the refining of data into a more compact, understandable and valuable form to humans, labeled as *"knowledge"* [53] (see fig.2.2)- to highlight the importance of the quantity and quality of the knowledge extracted from the raw data [9], because in business knowledge is *"money"*.

This BI dedicated analytics process, illustrated in fig.2.3, aims to help the decision makers to maximize the financial gains of their businesses. Thus, each step of the process is carried out by a class of specialists in order to guarantee its success. The importance of the specialist grows higher as we get close to the top, and the number of specialists of each class decreases similarly, following this progression:

1. **Data Base Analyst**: this class is in charge of the *data gathering* from several sources (data bases, web logs, individuals transactions...), the *data pre-processing* as defined in the original KDD, and the *data exploration* which is the equivalent of the *transformation* step of KDD.

2. **Data Analyst**: or *Data Miner*, discovers useful information from the previously data.

3. **Business Analyst**: presents the discovered information, in an understandable manner to the end users, through appropriate visualization techniques.

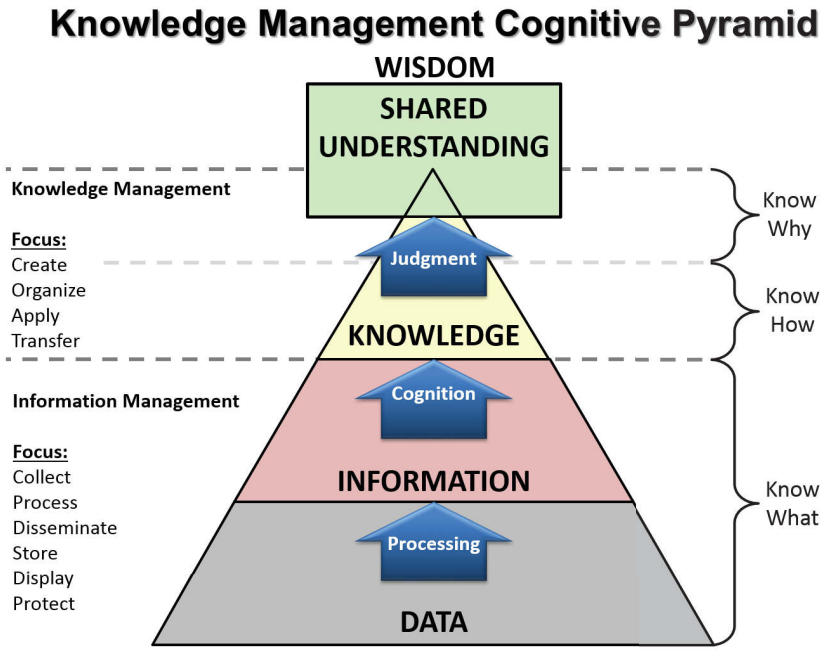

Figure 2.2 — The knowledge pyramid [53].

4. **End User**: or *Decision Maker*, has the most important and final role. Since they are at the top of the BI analytics process, the end users make their own knowledge on the data and plan a strategy to boost their businesses accordingly.

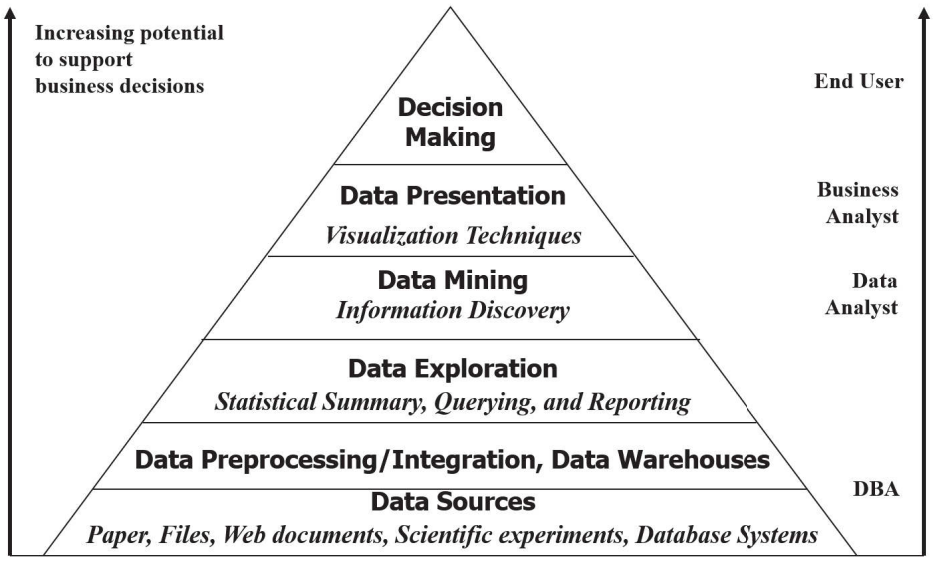

Figure 2.3 — Business Intelligence workflow [54].

2.1.3 KDD in Machine Learning and Statistics

Machine Learning (ML) refers to the analytics process, illustrated in fig.2.4, of teaching computers how to automatically extract important information from examples to achieve improved prediction or search capabilities for associations and/or patterns in data [55]

Based on the goal of data analytics, machine learning algorithms are organised taxonomically in two major categories. The first one is the *"supervised"* learning algorithms that build models from data labeled with their known output, to be used for predicting the output of unseen data or new data that fit the learned model. The second category of learning algorithms is known as *"unsupervised"*. They use unlabeled data to reorganise them or extract patterns.

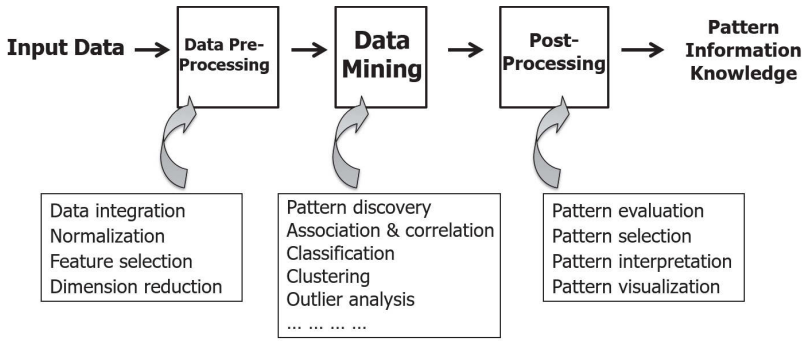

Figure 2.4 — Data analytics in Machine Learning and Stats community [54].

In addition to the work of designing new and more efficient machine learning algorithms, the ML-Stats community focuses as well on pre-processing and post-processing.

The former step aims to get the data ready for processing, by changing the shape, the size and data, while trying to preserve at best their integrity, in order to improve the relevance of the machine learning output and at the same time decrease its needs of computational power and processing time. The latter refines and displays the models and patterns learned to help the user in building his knowledge on the data.

Now let us summarize the previous definitions to get an unique one, on which we can rely for understanding what comes next in this document.

2.1.4 Summary of KDD

Basically *"Data Analytics"* designates analytical processes that take raw data as *input*, then *analyze* them and finally *output* the analysis results (see fig.2.5).

Moreover, each of the three main steps of data analytics contains at least one task that takes part in transforming raw data into knowledge. *Gathering* raw data is the mandatory task of the *input* step.

Furthermore it is mandatory for the whole data analytics process, because without data there is no reason for the data analytics to be used. For the *analysis* step, *data mining* is the only task and the most important globally. *Visualization* is the compulsory task of the last step of the data analytics process, the *output*, to allow the user to understand the information

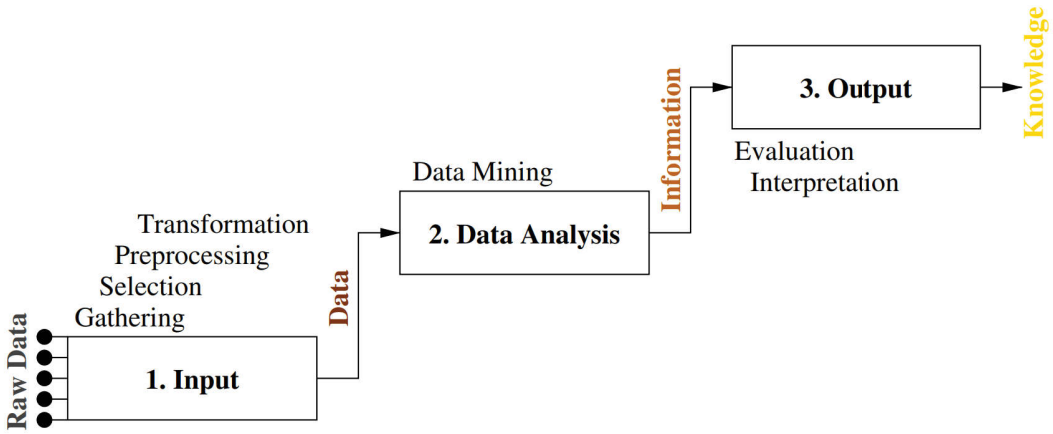

Figure 2.5 — Summary of Data Analytics [56].

and build his own knowledge.

To conclude, one must know that commonly, extra tasks are added to the basic data analytics process, in order to improve the efficiency[4] of each step and thereby the overall process.

2.2 Data Analytics Elements

In this section, the data analytics processes are broken down into their basic elements, the *"tasks"*, which can be combined in several ways to build different data analytics processes. These data analytics elements are organised following their respective data analytics steps.

2.2.1 Input: Pre-Processing

As said before, collecting *"raw"* data is the first and primordial task of any data analytics process. These collected data can be transformed [57] to ease and enhance the work during the next steps.

2.2.1.1 Data Gathering & Storage

Gathering and storing data is mandatory, considering that data can be of various origin and volatile, meaning we can not assume to have neither the same structure among all the sources nor a permanent access to them. Therefore, we classically have to keep the collected data in a same place with a uniform structure. As a result, the *Data Bases*, and by extension the *Data Warehouses*, were developed to provide a safe storage capability.

- **Data Bases (DB)**: the data base technology has evolved through time, starting from the hierarchical data bases, network data bases to relational data bases [58], which rely on relational algebra to store the data and the relationships between them.

[4] Relevance and computational complexity.

Besides providing a safe shared storage, data bases have two advantages that allow their users to manipulate the data, using a *Data Base Management System (DBMS)* through a *Structured Query language (SQL)* [59], without considering neither their physical organisation (physical independence), like knowing the files path or managing the file index and so on, nor how other users might manipulate the data in their applications, thanks to the independent *views* of the data (logical independence).

- **Data Warehouses (DW)**: the growing interest of the enterprises about data bases technology and their continuous need to boost their productivity via data bases improvement, had led to the birth of new data bases dedicated to decision support, known as *"Data Warehouses"*.

  The data warehouses are used for storing operational but also historical data, and for analyzing these data in complex dashboards and reports [60]. The data warehouses contains data that usually come from transactions data, which are mainly used for *OnLine Analytical Processing (OLAP)* operations to provide reports data through multi-dimensional queries.

2.2.1.2 Data Cleaning

Data Cleaning is strongly recommenced, since the collected data are quite often noisy. Hence, this task relys on some or all data cleaning techniques to fix noise in the data. Hereafter some examples of such techniques are provided.

- **Normalization** is a basic numerical operation that changes the scales of the data dimensions[5], in order to consider the data dimensions in a same fashion and avoid inappropriate conclusions [61].
- **Smoothing** aims to remove or change deviant[6] data values [62][63], caused by human and/or machine fault, for the sake of bringing the data closer to their true nature and thereby decreasing the occurrence probability of errors and misinterpretations during the following data analytic steps.
- **Fill in Missing Values** can be automatically, by means of for example *binning*, *regression* or *clustering*, to predict the most probable value to impute to the missing data [64][65]. It can also be done interactively with the help of a human who chooses the right value.

2.2.1.3 Data Compression

This optional task, addresses the issue of the data size by reducing it [66], while maintaining the data integrity.

- **Horizontal Compression** is the most applied data compression, which purpose is to reduce the number of data features, and results from the *Curse of Dimensionality*. It occurs when the number of features is high and the number of data instances[7] is not

[5] Also known as *"attributes"*, *"features"*, *"variables"*, *"columns"*.
[6] Far from what or where it should be.
[7] Also called *"tuples"*, *"records"*, *"lines"*.

high enough for the mining algorithm to distinguish between the dimensions.

This task is done by *"Dimensional Reduction"* [66] algorithms, like the *Principal Component Analysis (PCA)* [67], and/or *"Feature Selection"* [68] [69].

- **Vertical Compression** is less frequent but useful. It reduces the number of data instances thanks to *"Data Sampling"* [66] and/or *"Instance Selection"* [70].

Before going to the next data analytics step, one should know that, firstly, there is some exceptions or special tasks of data pre-processing, like the *Data Science Machine* [71], where new features are generated in order to find new relations in the data set. Secondly, the combination of the data cleaning tasks, which tasks to apply and in which order, is important for the results relevance and depend on the data analytics process objective [72].

2.2.2 Analysis: Data Mining

Data mining algorithms are categorized, based on the data mining purpose, in *descriptive* mining and *predective* mining [11][54].

2.2.2.1 Description

Description focuses on finding human-interpretable patterns describing the data.

- **Data Characterization and Discrimination**: the goal behind this task is to build a profile of some data instances of interest (characterization), according to some criteria given by the user, and understand how these instances are different from others (discrimination). This task is mostly related to data warehouses exploration [73].

- **Frequent Patterns Mining**: when a structure, that embodies relationships amongst the data, is repeated over the data instances, it is called a *"frequent pattern"*. Finding such patterns is a data mining task that goes by iteratively assembling and disassembling the data in subsets, in order to find and check which data subsets better represent frequent patterns. These patterns are: *itemsets* for transaction data, *subsequences* for historical or time related data, or *substructures* related to data graphs. These patterns are usually used for inferring *association rules* [74], which express structural and/or temporal relations between the data.

- **Clustering**: one of the most spread and known data mining task, clustering consist in finding sets of data instances that are very much alike. In other words, clustering is bringing *similar* data instances together in sets called *clusters*, based on a data similarity metric that usually combines the data attributes in some way.

  The existing clustering algorithms differ from each other, mainly, on how they make this attributes combination [75]. Thus, there is partitioning algorithms (*K-Means* [76], *K-Medoids* [77]), hierarchical algorithms [78], density-based algorithms [79], and grid-based algorithms [80].

2.2.2.2 Prediction

Prediction involves using, some or all, data attributes of a database to *predict* (forecast) unknown or future values of other attributes of interest.

- **Classification**: as the name suggests, predicts the *"class"* or the category to which data instances belong, typically by means of a supervised machine learning algorithm. They use a *training* set of *labeled* data. It contains some data instances (examples, observations) with their class (label).

  A classification algorithm builds a predictive model that encapsulates the wanted classification function. The predictive model is learned as follows: first, a random model with the available data features and the class feature (target), is set. Then, the examples are presented one by one to the learner[8]. Each time the learner makes the required modifications of the model seeking to minimize the model prediction error, given by an *error* function that computes how much the model prediction is far from the real target.

  The classification algorithms [81] differ on how they define their error function and how they update their model to correct its prediction. To achieve a classification task, it is common to use a *Support Vector Machine (SVM)* [82], a*Feedforward* and *Back Propagation Artificial Neural Network (ANN)* [83], a *K-Nearest Neighbors (KNN)* algorithm [84], a *Decision Tree* [85] and by extension a *Random Forest* [86], or any combination of classifiers[9] known as *"ensemble learning"* [87].

- **Regression**: classification and regression are alike with only one distinction : the classification predicts discrete values, whereas the regression predicts continuous values for the class. We can see the regression as a mathematical function, or a model more generally speaking, that translates or maps the data trend in order to be used to predict future and missing values.

  Since classification and regression are similar, the regression algorithms [88] are commonly classification ones with some, or no, updates to deal with the continuous nature of the target class, like *General Regression Neural Network (GRNN)* [89], *Support Vector Regression Machines* [90].

- **Dependency modeling**: this task focuses on discovering how variables (data features) depend on each other at a *structural* level and a *quantitative* level. The former exhibits the causal relationships through the data features. The latter specifies the strength of these relationships. The dependencies constitute a *beliefs network*, which is used for inferring domain expert knowledge [91].

  Beliefs networks are of two kinds: *probabilistic* networks [92] or *possibilistic* networks [93]. The probabilistic networks rely on the probability theory to model the dependencies, mainly by means of the *Bayes* theorem leading to *Bayesian* networks [94]. The *possibilistic* networks turn to a more flexible theory, the *possibility* theory, which is better suited to deal with uncertainty.

[8] An instance of one machine learning algorithm.
[9] Classification algorithms.

- **Outlier Analysis**: also called *"outlier detection"* or *"anomaly detection"*, consists in finding *abnormal* or *deviant* elements compared to the whole database [95][96]. The outliers, or anomalies, can be *punctual* or *collective*. The punctual outliers, as the name suggests, are one point anomalies which can be detected *globally*, considering the deviation from all the data, or *contextually* relatively to a sub data set of interest called a *context* or *profile*. The collective or longitudinal outliers are anomalies only if considered as a group and not individually.

Although the boundaries between prediction and description are not sharp, because some of the predictive models can be descriptive, to the degree that they are understandable, and vice versa, the distinction is useful for understanding the overall discovery goal.

2.2.3 Output: Post-Processing

Since it is the last step of the data analytics processes, the *output* step allows the users to produce their own knowledge based on the results of the previous step.

2.2.3.1 Visualization

Visualization is the basic task of the output step. It converts to mined[10] patterns and the learned models in a more human-readable form, thanks to numerous techniques [97]. The reason behind the abundance of visualization techniques is the difficulty to have a highly exhaustive representation of the mined[10] information, while exhibiting the most peculiar or potentially interesting information, especially when dealing with multi-dimensional data.

2.2.3.2 Evaluation

It is common for the data mining algorithms to produce a lot of information. Consequently, when information is presented to the users, they may struggle to identify which parts are, the most, relevant ones. Therefore, the mined information must be evaluated with *interesting* measures, that combine novelty, usefulness, and simplicity. The choice of these measures depends on the nature of the data mining task [98][99][100].

2.2.3.3 Annotation

After identifying *interesting* or *relevant* information, data analytics users may want to save or share their knowledge in order the help, themselves or others users, to interpret the meaning of the mined information the next time they analyze the same or related data. This task is known as *semantical annotation* [101]. Moreover, when these annotations are well done and structured, usually by using a semantic network called *"ontology"* [102], they can be used to automatically answer users questions about the domain data [103].

[10] The output of a data mining algorithm.

2.3 Limitations of Traditional Data Analytics

Tsai et al, in their survey of Big Data Analytics [56], have quite rightly pointed out the main limitations of data analytics, when faced with Big Data. According to their observations, most data analytics processes have limitations for big data, that can be described as follows:

- **Unscalability and centralization**: most of the data analytics processes are not for large-scale and complex data sets (big data). Traditional data analytics cannot be scaled up because their design does not take into account big data. Their design typically assumed they will be performed on a single machine, with all the data in memory. For this reason, the performance of traditional data analytics will be limited in solving the volume problem of big data.
- **Non-dynamic**: ordinarily traditional data analytics cannot be dynamically adjusted for changing situations, meaning that they do not analyze the input data on-the-fly. The extracted patterns and models are usually fixed and cannot be automatically changed. As a result, the performance and iterative nature of traditional data analytics may not be useful to handle the fast generation of big data.
- **Uniform data structure**: generally data analytics assumes that the format of the input data will be the same during the analytical process. Therefore, traditional data analytics may not be able to deal with the changing of formats of the input data.

To conclude this chapter, we can assert that tremendous works have been achieved, by various communities, for knowledge discovery from data. Furthermore, since there are no formal rules to choose neither the right tasks, from all of the data analytics tasks, nor the best way to combine them to get the best knowledge, the data analytics users rely on their experience, analytical skills and domain expertise in order to make the best choices. Consequently, there is several data analytics processes, each one dedicated to a specific domain or global task [104].

Also, because the traditional data analytics were not designed for large scale and complex data, they are almost unfit for analyzing big data. Redesigning and changing the way the data analytics process works are two critical trends for big data analytics. Several important concepts in the design of big data analytics processes, and their implementations, will be given in the following chapter.

3 Big Data Analytics: from Theory to Practice

As defined in chapter 1, data science aims at developing the set of technical and analytical tools which, put together, constitute the *Big Data Analytics*, to analyze big data. However, this is a recent statement that evolved and has been extended from the classical data analytics, presented in the previous chapter. Because of the evolution of data to big data, this called for an evolution of the data analytics processes on the conceptual level as well as on the technical level.

In this chapter, one can follow how the evolution of the knowledge discovery has taken place progressively, leading to the current state of big data analytics. Thus, the focus is given to big data analytics, as the application domain of this thesis. I present the origins, the current definition, architecture and implementations of big Data Analytics.

First, let's see how the *Knowledge Discovery from data (KDD)* process has changed in order to stress the importance of redesigning the KDD steps. Since data analytics is also designated as knowledge discovery from data, then we can call the big data analytics as knowledge discovery from *big* data.

3.1 From Data Analytics to Big Data Analytics

The explosion of the available amount of data and the rise of their complexity[1] have led the conventional data analytics to their limits. Therefore, a lot of works have started to induce a shift of the data analytics paradigm in order to handle big data.

For the sake of a better understanding of the shift in the data analytics paradigm, a reformulation of the data analytics pipeline, illustrated in fig.3.1, is given as follows:

1. The data analytics is a straightforward pipeline that takes all the raw data to be analyzed as an input, which causes a huge bottleneck at the beginning of the pipeline.
2. Once the the first step is done, a huge and unique data stream is processed following the next steps of the pipeline.
3. Then, when the data analytics process is finished, the knowledge extracted can be injected as another input into the next iteration of the pipeline.

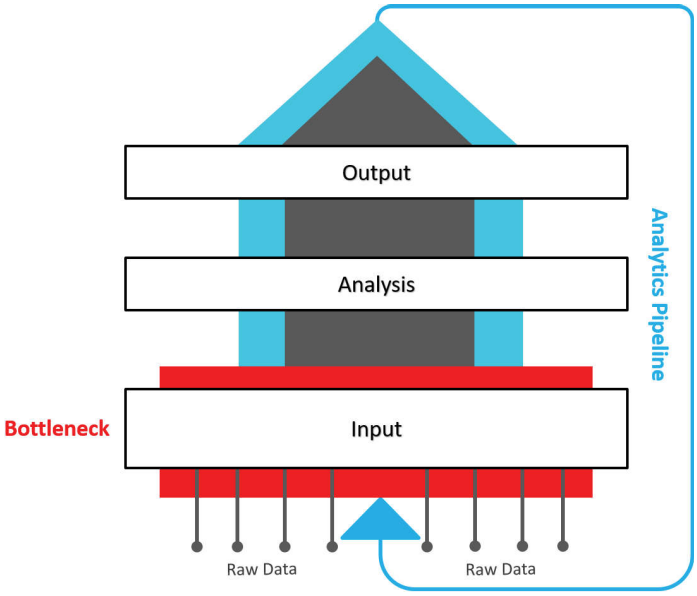

Figure 3.1 — Data Analytics pipeline.

Considering that the data analytics pipelines deal with the data as one big stream, the pipelines must be widened in order to process bigger data stream. This widening is done by using bigger storage depository, faster communication networks and more powerful computational units.

[1] Data dimensionality and variety.

Nonetheless, as a result of the technological limits, such widening is necessarily insufficient and new solutions have to be considered to allow bigger data to be processed.

Thence, a big data analytics pipeline should be flexible and able to be virtually extendable to handle the continuous growth of data. To do so, the analytics pipelines have to be upgraded and redesigned with respect of two constraints or goals:

1. The first one, is to reduce the overall bottleneck, by scattering it over each step of the analytics pipeline.
2. The second one, is dividing the data stream into several sub-streams. Then, processing them individually and merging them when necessary.

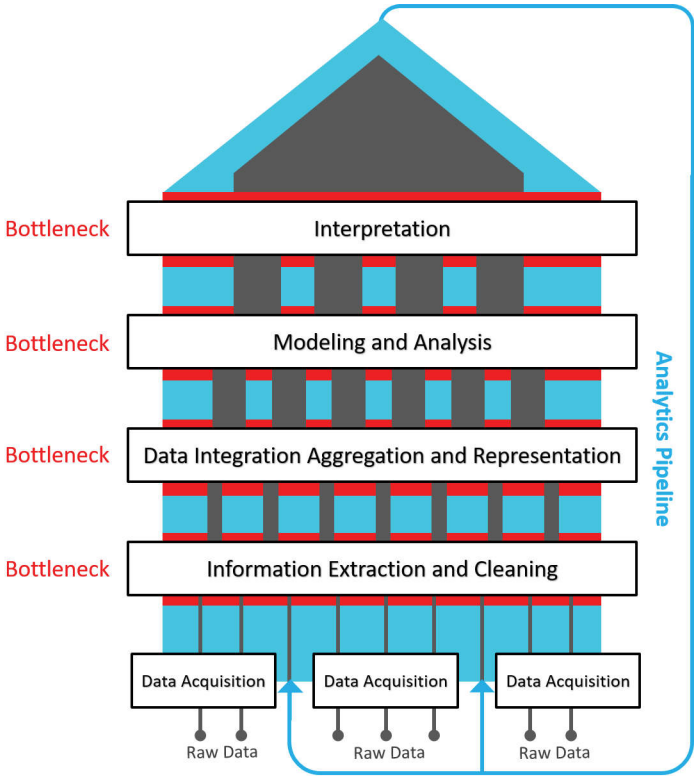

Figure 3.2 — Big Data Analytics pipeline.

As a result, the new analytics pipeline adapted to big data, depicted in fig.3.2, is defined as a data analytics pipeline that:

- ▷ breaks up the former input step into three preprocessing steps [105], in order to highlight their huge relevance in reducing the size and complexity of the input and thereby to globally ease the analytics process.
- ▷ improves each step, and by extension their component tasks, by means of *distribution*, *mobility*, *parallelism* and *collaboration*.

Hereafter is presented the new big analytics steps and the technical and/or conceptual improvements that allow these steps to handle big data.

3.1.1 Data Acquisition

The revolution base of big data analytics came from the democratization of data generative devices, called *"sensors"*, that record the state of the cyber-physical world[2].

Furthermore, every apparatus of the daily life, like housing appliances, cars, and even clothes and other wearables, are now starting to include sensing and recording capabilities. The set of all these sensors and *connected* devices, which can be mobile, form a huge network called *Internet of Things (IoT)*. IoT is also at the core of smart cities [106].

Consequently, the available amount of data is exploding, meaning their volume is growing, their generation speeding up, their dimensionality and heterogeneity increasing.

These big data can't be managed by conventional storage systems, like data bases and data warehouses. Therefore, new flexible storage technology is required in order to go with the flow of *IoT* big data. As a response, three major storage technologies have emerged.

3.1.1.1 Mobile Data Bases

Mobile data bases are data bases dedicated to mobile devices (laptops, smart-phones, connected wearables...) and insure a shared access to their respective data, thanks to the *Peer-to-Peer (P2P)* file sharing protocol over mobile ad hoc networks [107], even if the mobile devices are out of range or disconnected [108].

Mobile data bases require new mechanisms for data access and update. For example, Pitoura and Bhargava proposed an agent-based framework for providing consistent and recoverable access to heterogeneous mobile databases [109], because the multi-agent paradigm is well suited for dealing with concurrency and heterogeneity.

Despite the considerable improvements in data management, brought by this new kind of data bases, new issues have also been engendered [110][111]: concurrent data modification, data cashing and replication, huge network communication workload, increasing energy consumption and reducing the computing power of the devices.

Thus, a new type of data bases was proposed in order to try to keep the advantages of mobile data bases while overcoming their weaknesses.

3.1.1.2 Distributed Data Bases

Since it is hardly possible to store all the data in a single place or retrieve them directly from their mobile sources on demand, a new sustainable storage solution, *Distributed Data Bases*, was designed as a trade off between fully centralized storage and fully mobile storage [112].

Distributed data bases allow to store data over several scattered depositories, for the sake of providing fast and consistent access to the data, wherever the users are, thanks to mainly three major internal data management proprieties [113]:

- *data partitioning (Partition)*: dividing the whole database in several parts, called

[2] The real and digital worlds.

partitions, depending on the number of available depositories. From one perspective, it is a good aspect of the distributed data bases that supports their extension when the data grow. From another perspective, it increases the number and the cost of data joins[3] to retrieve some information scattered over several partitions, especially when they are physically far from each other (the network communication cost is added).

- *data duplication (Availability)*: virtually, the intersection of all data partitions is empty, but in reality it is not as a result of the data duplication. Data duplication is a safeguard mechanism that copies the data from one partition to one or more other partitions.

  The duplication provides virtual data backups when a partition is not accessible anymore, due to a failure of the related depository or a congestion of the communication network. The duplication is said to be good when the users can access the whole or most of the data base even if one or few partitions are inaccessible.

- *data replication (Consistency)*: is related to the data duplication, because when some modifications are done on some data they should also be done, *replicated*, on the duplicated data.

In fact, it is impossible for a distributed data base to achieve all the three proprieties (Consistency, Availability, Partition) at the same time. This is known as *Brewer's CAP theorem* [114]. However, this statement is nowadays not so true, because by explicitly handling partitions, designers can optimize consistency and availability, thereby achieving some trade-off of all three [115]

3.1.1.3 NoSQL Data Bases

Admittedly, the distributed data bases have shown their usefulness at the beginning of the big data age. Nevertheless, more and more new big data does not conform to a rigid relational schema. Hence, the users can't be bound by the structure of a *Relational Data Base Management System (RDBMS)* and need something more flexible with high performances.

As a result, a new data management paradigm, the *Not only SQL (NoSQL)* [116], was created. NoSQL data bases [117], are non-relational distributed data bases that:

- provide high performance by storing massive data without explicit and structured mechanisms to link data to one another, using new data structures (key-value, column, document, or graph). Choosing the suitable structure, depending on the big data analytics goal, makes some operations faster.

- allow flexibility, thanks to simpler horizontal scaling (defined in the next section) to clusters of machines, which is problematic for relational databases.

In return, they compromise their consistency. Instead they offer *"eventual consistency"* [118]: modifications are *"eventually"* propagated to all storage nodes (depositories) over time.

[3] Operations to combine data from two sets of data (i.e. two tables).

Hence, queries for data might not return *up-to-date* data immediately or might result in reading data that are not accurate, which is known as *stale reads*.

In addition, NoSQL databases have several inconveniences: no SQL support which is an industry standard, lacking of transactions, reports and other additional features, not mature enough for most of the NoSQL database products that were created in recent years and so on.

3.1.2 Information Extraction and Cleaning

As the volume, the velocity and the complexity of data grow, the proportion of poor-quality data (noisy and untrustworthy) grows as well; leading to extract poor-value knowledge. In other words, *"Bigger Data are Not Always Better Data*.

Therefore, the design of big data analytics needs to consider how to reduce or get rid of the noisy data as well. Tang authored a survey of automatic methods for big data cleaning [119].

Nonetheless, decisions about what attributes and instances will be processed and which will be ignored is, to some extent, an inherently subjective process and computationally expansive. So, relying on humans skill is one way to deal with it.

Thence, some works came up with users interactive solutions for big data cleaning, like Crowdcleaner: data cleaning for multi-version data on the web via crowdsourcing[4] [120]; and some works study quantify the value of user-level data cleaning [121].

3.1.3 Data Integration, Aggregation, and Representation

These concern the problem of handling a vast quantity of data that the system is unable to process. Thus, preprocessing is an important task to make the big data analytics pipeline able to handle the input data.

A portion of the big data analytics studies still focuses on how to reduce the complexity of the input data, because even the most advanced computing technology cannot efficiently process the whole input data by using a single machine in most cases.

3.1.3.1 Sampling and Compression

Sampling and compression are two representative data reduction methods for big data analytics, because reducing the size of data makes the data analytics computationally less expensive, thus faster, especially for the data coming to the system rapidly and massively, from *IoT* for instance.

In addition to make the sampling data representative of the original data effectively [122], it is necessary to define how many data instances need to be selected for a specific data mining method. But it is another research issue [123], because it will affect the performance of the sampling method in most cases.

[4] Divide work between participants to achieve a cumulative result.

3.1.3.2 Performance Optimization

Performance optimization is another issue for big data reduction methods. For this reason, Zou et al. [124] proposed a promising solution for efficient big data reduction, by means of clustering the input data to divide them into several different groups and apply an optimized compression algorithm on these input data according to the clustering information.

Moreover, by using domain knowledge to design the preprocessing operator it is possible to handle the high computations of preprocessing solutions for the big data. For instance, Dawelbeit and McCrindle [125] designed an elastic pre-processing framework to divide the input data between the computing processors.

3.1.4 Modeling and Analysis

Most available data mining algorithms are not very suitable for big data mining applications as their design is based on limited and well defined data sets [126].

Thus, big data analytics need to implement advanced and more sophisticated data mining algorithms to deal with big data efficiently. These algorithms need to be designed as distributed components across various geographical locations and need to work effectively across heterogeneous environments; and be capable of managing and operating in highly dynamic environments.

The primary cornerstone behind the big data mining algorithms is *parallelism* or *parallel computing*. Parallelism is the simultaneous execution of several processes and computations, using more than one processor (CPU) be it on a single machine or on networked machines [127], in order to:

- either *speed up*: decrease the time needed for carrying out all the computing tasks, called *response time*;
- or *scale up*: handling larger tasks by increasing the degree of parallelism, known as *throughput*.

Several studies, like [128] and [129]), attempted to modify the traditional data lining algorithms to make them work on a parallel computing environment or to develop new algorithms which work naturally on a parallel computing environment, as shown in fig.3.3.

However, there is a price attached to the benefit of parallelism, namely the *synchronization* cost. Synchronization is the mechanism that makes one of the parallel processes wait when they try to concurrently access to the same unshareable resource, leading sometimes to a *deadlock*[5].

Usually in parallel data mining, the synchronized resources is the data set. So, to avoid the parallel processes synchronization and reduce the risk of deadlocks, the parallel data mining algorithms have came to distribute the data over the computational units (CPUs, machines), normally by means of the distributed data bases.

[5] A state in which some processes wait for each other to take action.

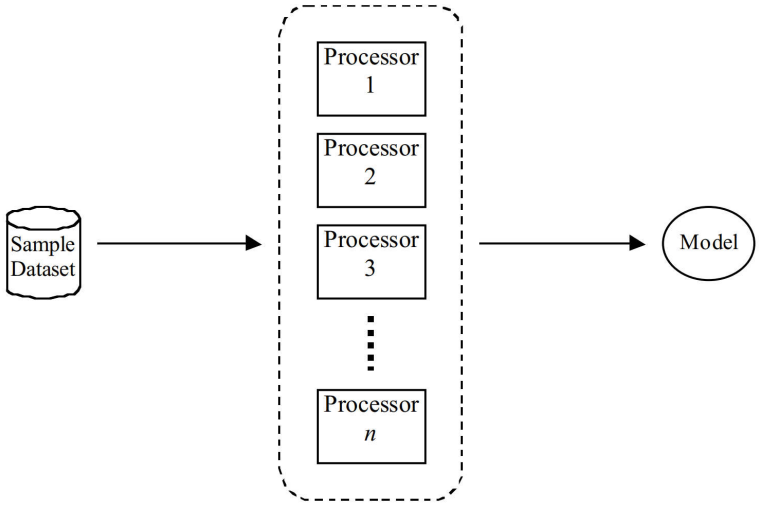

Figure 3.3 — Parallel data mining [128].

This consequently engendered the distributed data mining algorithms [130][131][132], whose architecture is illustrated in fig.3.4.

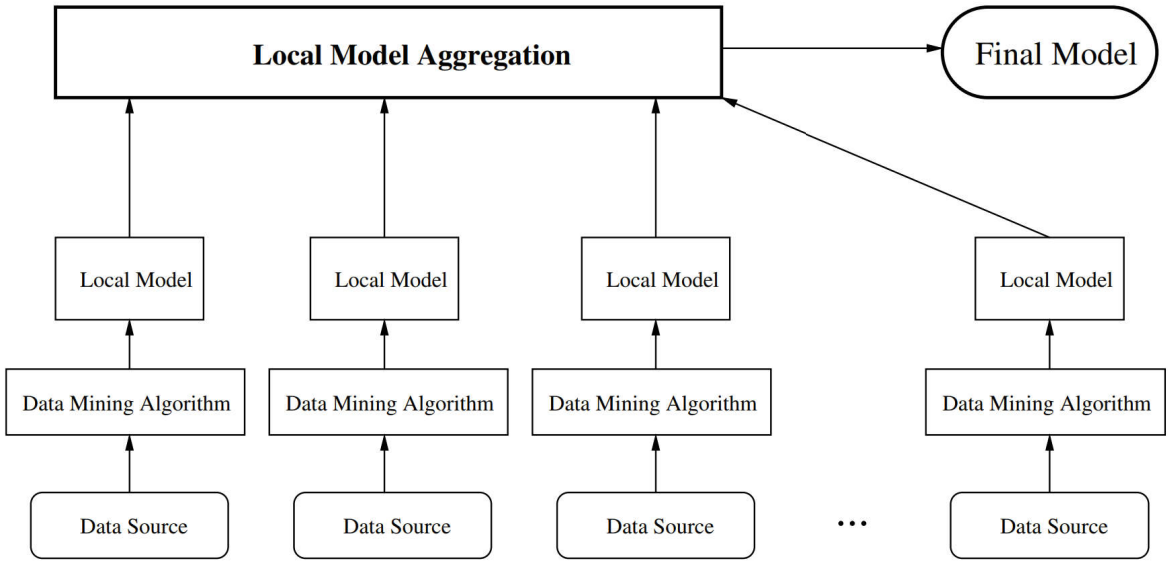

Figure 3.4 — Architecture of distributed data mining algorithms [133][56].

To sum up, big data analytics implies having data mining algorithms that are linearly scalable, able to handle high throughput multi-formatted data, with a high degree of parallelism and a distributed data processing.

Finally, to know what type of data mining can be applied, according to available computing and storage architecture, one can rely on table 3.1.

[6]Conventional data mining applied on classical data bases.
[7]Conventional data mining applied on distributed data bases.

CPU \ Data	centralized	distributed
mono	sequential mining[6]	centralized mining[7]
multi	parallel mining	distributed mining

Table 3.1 — types of data mining algorithms

3.1.5 Interpretation

Like the previous steps of the big data analytics pipeline, the last step of the pipeline, the *interpretation*, also is affected by the size, the velocity and the complexity of the data. The quantity of mined information form big data requires new tools to help the users to interpret them, and thereby infer their own knowledge, individually and collectively.

Again, similarly to the other big data analytics steps, the knowledge extraction tools from big data have evolved mainly by improving their performances. For instance through parallelism for visualization and crowdsourcing for annotation.

3.1.5.1 Visualization

Big Data visualization is not that easy like traditional visualization of relatively small data sets. The extension of traditional visualization techniques has already begun but still has progress to do.

When it comes to large-scale data visualization [134], most techniques use data compression and give a good approximation to large-scale data, such as feature extraction[8] and a geometric modeling[9] to significantly reduce the data size before the actual data rendering[10].

For more close and intuitive data interpretation, some researchers try to run batch-mode software rendering of the data at the highest possible resolution in a parallel way [135]. Choosing proper data representation is also very important when we try to visualize Big Data.

3.1.5.2 Collective Annotation

Even if a lot of researches are in progress to design big data visualization techniques that maximize the knowledge an individual can extract on its own, it is difficult if not impossible for one person to annotate and interpret all mined information.

When it comes to big data, interpretation would be better as a collective effort, either to distribute the work or to cross-validate several interpretations of the same information. The main mechanism for collective annotation is *crowdsourcing*: an online, distributed problem-solving and production model that has emerged in recent years [136].

[8] Select the best attributes that describe the data.
[9] Use different shapes and forms for intuitive data reading.
[10] Process of drawing the visualization output.

Amazon's Mechanical Turk [137] is one known example of crowdsourcing platform, where a *requester* can submit an annotation/interpretation task with some amount of money, as a compensation. Then, the platform divides the task and the compensation over several *turkers* who annotate/interpret their share of the data in exchange of a very small percentage of the compensation.

Besides, crowdsourcing is not only dedicated to information interpretation, it can also be used to collect data and/or features and meta-data to enhance the current semantics of data.

3.2 Big Data Analytics Platforms: Cloud Computing

In view of the technical evolution of the data analytics pipeline, the hardware and software architecture of computing platforms had to evolve too in order to support the big data analytics processes. These platforms are designated as *"clouds"*. In other words, the cloud refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [138].

Cloud computing is closely related to big data. Its main objective is to use huge computing and storage resources, commonly distributed over several computer clusters called *"commodity servers"* (readily available, all-purpose, standardized and highly compatible piece of hardware that can have various kinds of software programs installed on it), under concentrated management, so as to provide big data applications with fine-grained computing capacity [139].

The democratized use of the cloud computing platforms has altered the view of the big data analytics pipeline so much that it became the cornerstone of the big data analytics, as illustrated in fig.3.5).

Architecturally speaking, cloud platforms are basically composed of three layers: an *Internal Storage Layer* of the data to be processed, a *Computing Layer* used to process them, and an *Application Layer* which provides various analytics packages to allow the users to build their big data analytics process.

Each layer rely on the previous one and can be implemented in various ways, as described hereafter.

3.2.1 Internal Storage Layer: File Systems

Since on one hand the main drive of cloud computing is the need to get the data closer to where their are processed, and on the other hand the big data are usually stored in NoSQL databases (very big and distributed databases), the conventional internal file storage mechanisms known as *file systems*, which are the foundation of the applications at upper levels, had to be improved by means of files distribution over the cloud for the sake of a fast and fault-tolerant access to the data when needed.

As a result, new distributed file systems dedicated to big data on cloud platforms appeared.

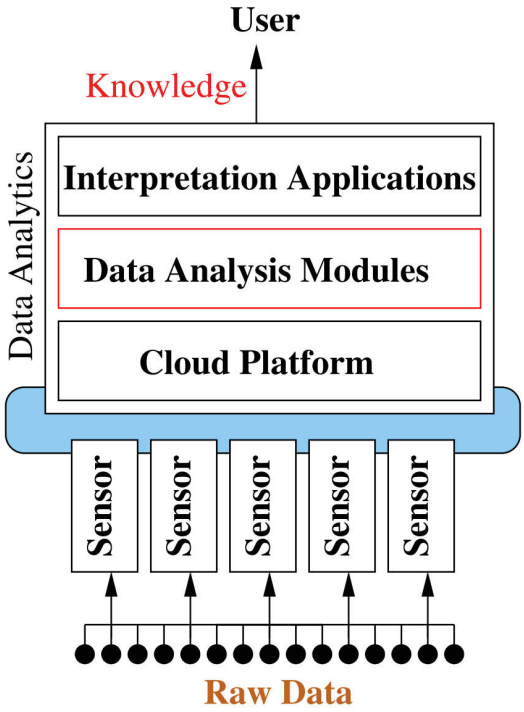

Figure 3.5 — The basic idea of big data analytics on cloud system [56].

- **GFS (Google File System)** [140]: is an expandable and consistent distributed file system that supports large-scale, distributed, data-intensive applications. GFS uses cheap commodity servers to achieve fault-tolerance and provides customers with high-performance services. However, GFS also has some limitations, such as a single point of failure and poor performances for small files. Such limitations have been overcome by *Colossus* [141], the successor of GFS.

- **HDFS (Hadoop Distributed File System)** [142]: as a derivative of open source codes of GFS, is the most widespread distributed file system designed to run on large clusters of commodity hardware. HDFS has many other goals, the major one is to detect and handle failures at the application layer. This objective is realized through a well-organised mechanism of replication.

- **Tachyon** [143]: is a fault-tolerant distributed file system, based on HDFS, which enables file sharing (data I/O) at high speed (system memory like) across a cluster, which is achieved by using memory more aggressively. Tachyon can detect the frequently read files and cache them in memory thus minimizing the disk access by different jobs/queries.

  Another advantage of using Tachyon is its support for raw tables. Tables with hundreds of columns can be loaded easily and the user can specify the frequently used columns to be loaded in memory for faster access.

- **Other Files Systems** [139]: Amazon came up with the *S3* file system to provide simple storage service [144]. Microsoft developed *Cosmos* [145] to support its search and advertisement business. Facebook utilizes *Haystack* [146] to store the large amount of

small-sized photos. Taobao[11] also developed its file system *TFS (Taobao File System)* [147].

In conclusion, distributed file systems have been relatively mature after years of development and business operation.

3.2.2 Computing Layer: Virtualization & Scaling

The computing layer is responsible of the efficient use for the extendable computational resources and rely mostly on two technological key concepts: *virtualization* and *scaling*.

3.2.2.1 Virtualization

In simple terms, virtualization allows an easy sharing of the same computational resources independently of their architecture [148].

More specifically in cloud computing, virtualization is an additional internal layer, between the hardware and the software, that make one or more heterogeneous systems (virtual machines) able to run at the same time on the same physical machine under the management of the *Virtual Machine Monitor (VMM)* [149], as shown in fig.3.6.

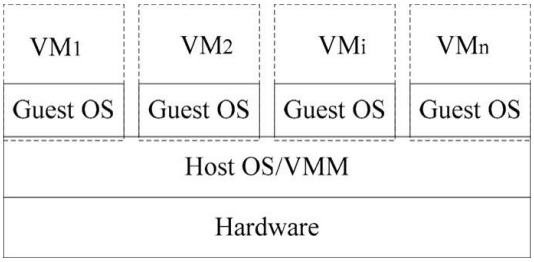

Figure 3.6 — Architecture of virtual machine system [150].

VMMs provide two technically attractive features [151], listed hereafter.

- ▷ *Encapsulation of virtual machines states*: facilitates the tasks of load balancing, virtual machine replication, storage and transport, suspend/resume scenarios, hardware and software failure handling.
- ▷ *Strong isolation between virtual machines*: besides the dramatic cost benefits, thanks to the transparent multiplexing of virtual machines over the same hardware, security and reliability advantages follow naturally. Indeed, a malfunctioning program, either due to bugs or security compromises, is isolated in its virtual machine and problems do not propagate to machines executing under the same VMM.

These advantages make virtualization highly useful for data-centric, multi-tenancy environments of big data analytics. Additionally, considering the fact that the productivity in developing future big data analytics software is an important cost factor, advances in this direction are contributing.

[11] The world's biggest e-commerce website and has over 580 million monthly active users.

3.2.2.2 Scaling

Scaling is the ability of expanding a system to handle the increased demands in terms of data processing [152][56]. To support big data processing, different platforms incorporate scaling in different forms.

From a broader perspective, the big data platforms can be categorized into two types of scaling: *vertical scaling (scale up)* and *horizontal scaling (scale out)*.

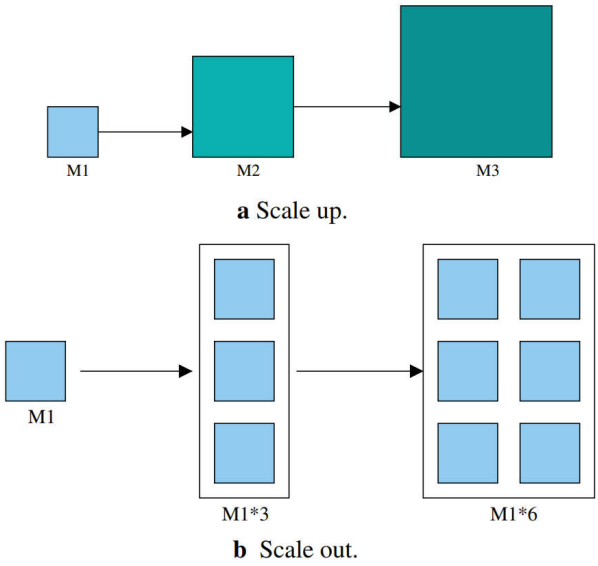

Figure 3.7 — The comparisons between scale up and scale out [56].

3.2.2.3 Horizontal Scaling

Horizontal scaling (scale out) is the distribution of the workload across many independent and heterogeneous servers or commodity machines added together in order to improve the processing capability.

for the scale out based system, all we have to do is to keep adding more similar computer systems to a system, in order to process bigger data, following one of these parallel/distributed programming models:

- **Peer-to-Peer (P2P) & Message Passing Interface (MPI)**: P2P is a decentralized and distributed network architecture (see fig.3.8) used as one of the first distributed computing models [153][154], whose scale out is practically unlimited. Millions of connected machines (peers) can provide as well as use computing resources and store data instances.

  MPI is the standard software communication paradigm [155] used in P2P networks to exchange data between heterogeneous peers [156]. Unlike other computing models, MPI based processes don't need to read the same data many times and all their parameters can be preserved locally.

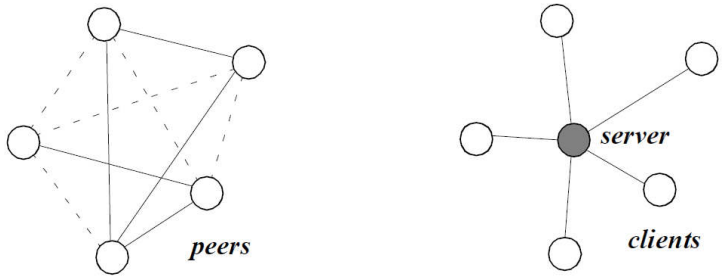

Figure 3.8 — P2P vs Client/Server architecture [153].

In addition, when MPI is deployed in a master-slave model, the slave machine can become the master for other processes. This can be extremely useful for dynamic resource allocation where the slaves have large amounts of data to process.

However, MPI has no mechanism to handle faults, thence a single node failure can cause the entire system to shut down. For this reason and with new computing models becoming widely popular, MPI is not being widely used anymore.

- **MapReduce** [157]: is a simple but powerful programming model suitable for processing semi-structured or unstructured data, pioneered by Google and developed by Yahoo! and other web companies, that rely on a divide and conquer strategy to break down complex big data problems into small units of work and process them in parallel. Then the solutions to the sub-problems are combined to give a solution to the original problem, as illustrated in fig.3.9.

  To do so, the MapReduce model relies on two functions or steps, both of which are programmed by users:

  - *Map*: at a high-level, mappers read the input data as *key-value* pairs from the file system, process them and generate some intermediate results as *key-value* pairs too and store them in the file system.
  - *Reduce*: reducers access the file system to aggregate the intermediate values, associated with the same intermediate key, to generate the final output, which is again written to the file system.

  The MapReduce programming model and its open-source version *Apache Hadoop* [158], which is based on HDFS, have been widely adopted in the big data analytics community for their simplicity, ease-of-programming, and combination of dynamic scheduling and selective re-computation.

  This lets the system automatically balance the loads on processors and readily tolerate failures through a deterministic replay mechanism, where a failed map/reduce task is simply re-executed [159].

  Nonetheless, MapReduce is by no means the universal solution to all parallel programming applications, and existing implementations have significant sources of inefficiency, due to some limitations:

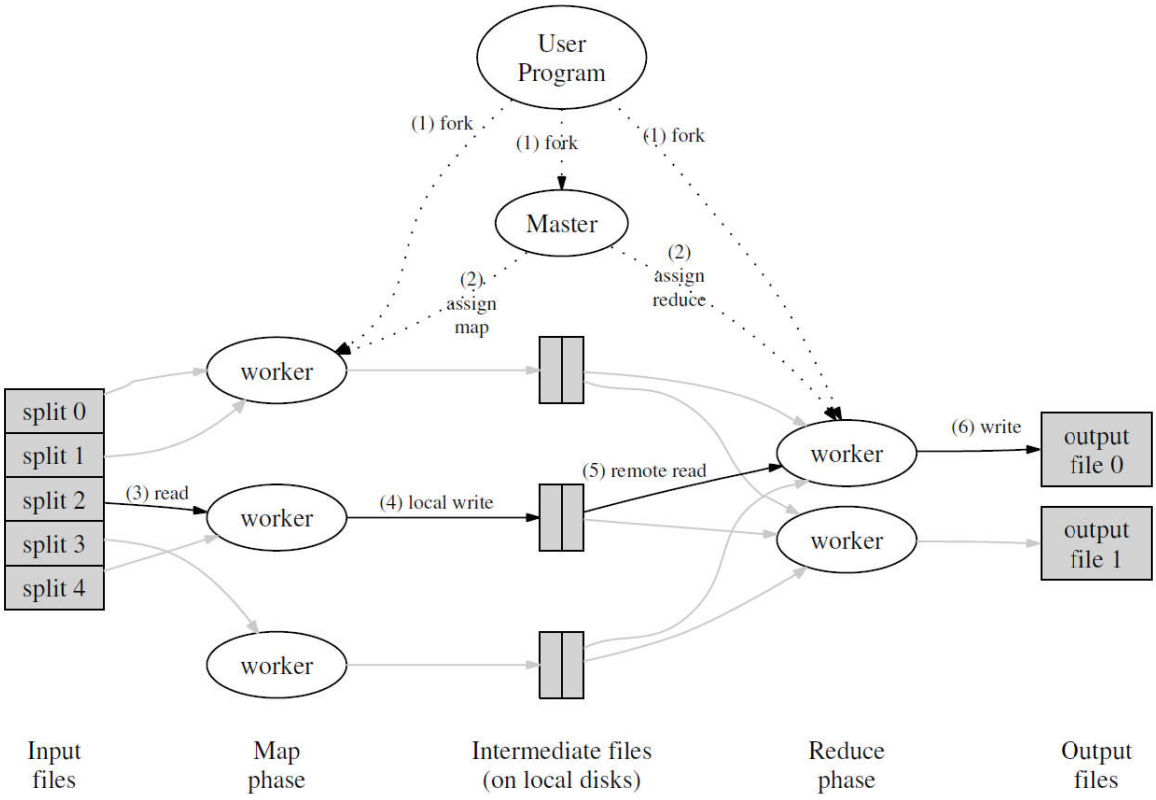

Figure 3.9 — Overview on a MapReduce execution [157].

– MapReduce is not designed for iterative processes. For each iteration, a new mapper and reducer have to be initialized and read the same data again and again from the disk, which significantly degrades the performance.

  Thus, a number of research efforts have aimed at improving this limitation of MapReduce, like *HaLoop* [160], *CGL MapReduce* [161], *Twister* [162], *imapreduce* [163].

– MapReduce requires large memory and storage for data replication and the whole system may be down when the master machine crashes for a system that has only one master. One solution to improve the performance is the design and use of next generation paradigms for big data processing, like *Apache Spark* [164] which relies on the high-speed file system Tachyon.

  Moreover, to solve these two problems, the *MapReduce Agent Mobility (MRAM)* [165] has been presented as a framework where each mobile agent can send its code and data to any other machine.

– MapReduce is inefficient in supporting simple SQL-like query operations, which usually involve just a few data elements, since MapReduce only provides two nontransparent functions, that cannot cover all the common databases operations, and requires performing computation over an entire dataset.

  The solution to this drawback is the last cloud layer, presented later in this section, that provides basic functions and advanced packages, which are typically hard to be maintained and reused, in order to provide various higher-level functionalities.

▷ **Dryad** [166]: while MapReduce restricts the data processing flow to map and reduce phases in favor of simplicity, Microsoft's computing model, *Dryad*, supports a more general data processing flow expressed as directed acyclic graphs [167] in which vertexes represent sequential programs (operation) written by the user and edges represent one-way data channels, as fig.3.10 schematically shows it.

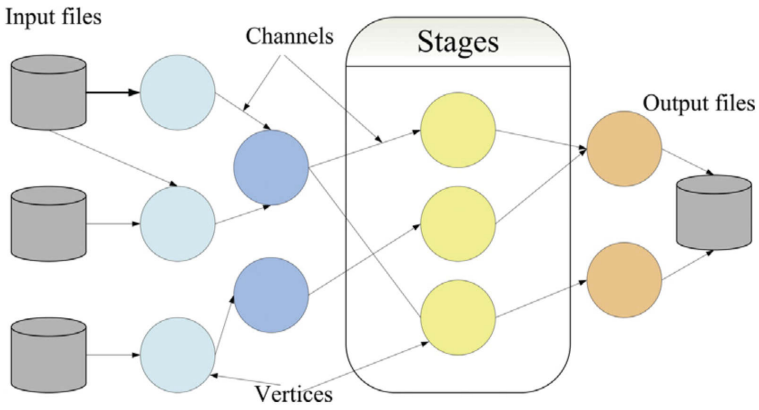

Figure 3.10 — The structure of dryad jobs [45].

Dryad executes the vertexes associated operations in clusters or workstations through network and transmits data via data channels. During operation, resources in a logic operation graph are automatically mapped to physical resources, as illustrated in fig.3.11. Additionally, Dryad can choose whether to store a data element as a file in a distributed file system, as a file in a local file system, or in a memory buffer.

Furthermore, Dryad allows vertexes to use any amount of input and output data, while MapReduce supports only one input and output set, and the generated graphs can also be updated after execution, in order to deal with unexpected events in the computation, like failure, or in response to user-defined policies decisions.

Because Dryad encompasses other computing models like Map/Reduce and the relational algebra, it is more complex and powerful in some degree.

The previous computing model was designed for batch processing. they are multi-purpose engines but not real-time engine. But stream data applications, such as processing log files, industry with sensors, etc. require real-time response for processing large amount of stream data.

In those applications, stream processing for real-time analytics is mightily necessary. Therefore, the real-time computing models are designed specially for real-time stream data analytics. Hereafter are presented two of such models:

▷ **Storm** [168]: is a distributed and fault-tolerant real-time computation system for processing limitless streaming data. It is also very easy to set up and operate, guarantees all the data will be processed, and provides competitive performances. Therefore, it has many applications, such as real-time analytics, interactive operation, on-line machine learning and continuous computation.

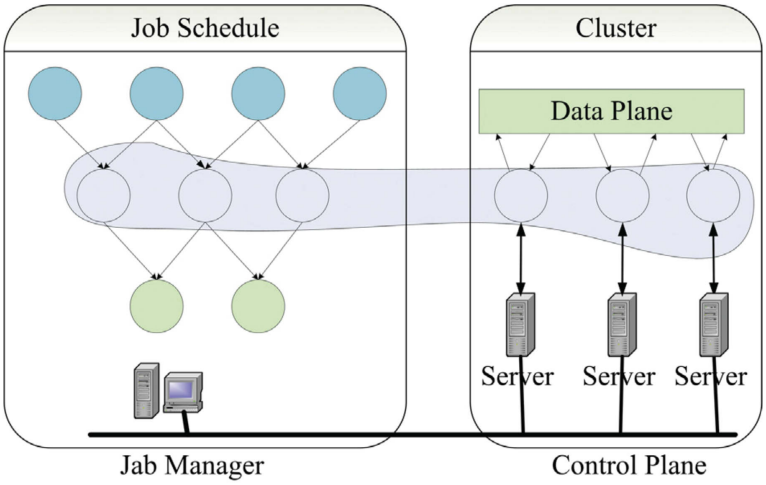

Figure 3.11 — Dryad architecture [45].

A Storm cluster is quite similar to a MapReduce cluster. However, on Storm users create and run different topologies for different Storm tasks. The key difference is that a MapReduce job eventually finishes, whereas a topology processes messages all the time, or until users terminate it.

A topology is a computation graph, as shown in fig.3.12, that can be created and submitted in any programming language. There are two kinds of nodes in topologies, namely, *spouts* and *bolts*.

A spout is one of the starting points in the graph, which denotes source of streams. A bolt processes input streams and outputs new streams. Each node in a topology contains a processing logic, and links between nodes indicate how data should be processed between nodes.

Therefore, a topology is a graph representing the transformations of the stream, and each node in the topology executes in parallel [45].

A Storm cluster consists of two kinds of working nodes. As illustrated in fig.3.13, there is only one master node and several worker nodes. The master node and worker nodes implement two kinds of daemons[12]: *Nimbus* and *Supervisor* respectively.

Nimbus is in charge of distributing code across the Storm cluster, scheduling works assigning tasks to worker nodes, monitoring the whole system. If there is a failure in the cluster, the Nimbus will detect it and re-execute the corresponding task.

The supervisor complies with tasks assigned by Nimbus, and starts or stops worker processes as necessary based on the instructions of Nimbus. The whole computational topology is partitioned and distributed to a number of worker processes, each worker process implements a part of the topology.

For Nimbus and the Supervisors to work swimmingly and complete the job fast, another kind of daemon called *Zookeeper* coordinates the system.

[12] a computer program that runs as a background process, rather than being under the direct control of an interactive user.

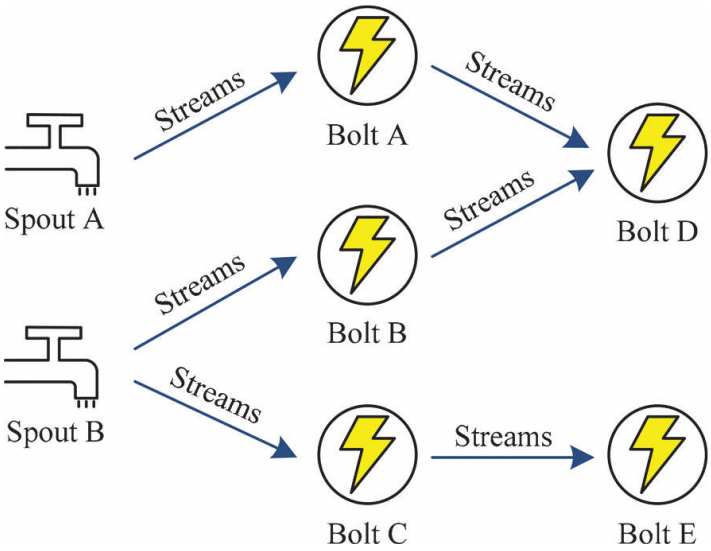

Figure 3.12 — Example of Topology [169].

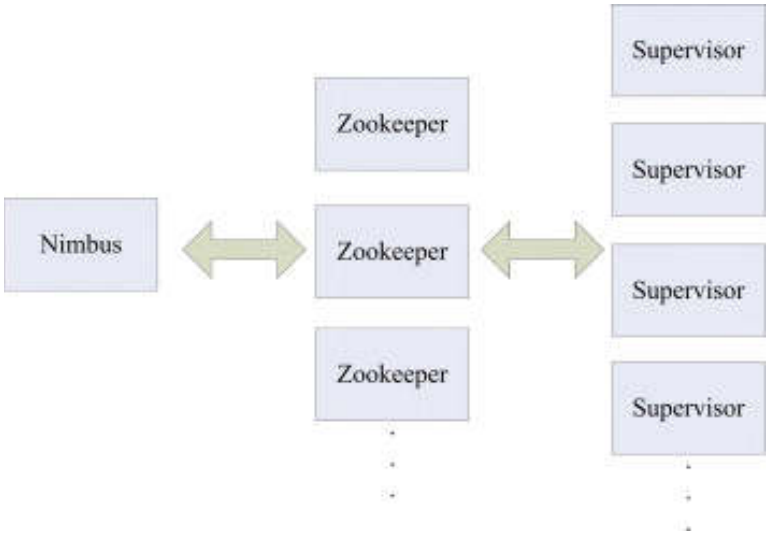

Figure 3.13 — A storm cluster [45].

▷ **S4** [170]: is a general-purpose, distributed, scalable, fault-tolerant, pluggable computing platform for processing continuous unbounded streams of data. It allows programmers to easily develop applications, and possesses several competitive properties, including robustness, decentralization, scalability, cluster management and extensibility [171].

The implementation of an S4 job is designed to be modular and pluggable for easily and dynamically processing large-scale stream data. S4 also employs Apache ZooKeeper to manage its cluster, like Storm does.

3.2.2.4 Vertical Scaling

From the perspective of platform performance, most of the traditional parallel processing models improve the performance of the system by using a new larger computer system to replace the old computer system, which is usually referred to as *"Vertical Scaling"* or *"scale up"*.

It is done by installing more processors, more memory and faster hardware, typically, within a single server and usually involves a single instance of an operating system. The most popular vertical scale up paradigms [152] are:

- **High Performance Computing (HPC) clusters** [172]: also called supercomputers, are machines with thousands of cores, powerful hardware which is optimized for speed, throughput, and failures resilience.

  Thus, the initial cost of deploying such a system can be very high. Furthermore, the cost of scaling up such systems is much higher compared to horizontal clusters and they are not as scalable, but they are still capable of processing terabytes of data.

  The communication scheme used for such platforms is typically MPI, which is not problematic in this case since these platforms don't require fault-tolerance mechanism.

- **Multi-core CPU** [173]: refers to one machine having dozens of processing cores. They usually have shared memory and only one disk (storage drive).

  The parallelism in CPUs is mainly achieved through multi-threading [174]. The task has to be broken down into threads. Each thread is executed in parallel on different CPU cores. Most of the programming languages provide libraries to create threads and use CPU parallelism. The most popular choice of such programming languages is Java.

  The drawback of CPUs is their limited number of processing cores and their primary dependence on the system memory for data access. System memory is limited to a few hundred gigabytes and this limits the size of the data that a CPU can process efficiently. Once the data size exceeds the system memory, disk access becomes a huge bottleneck. Even if the data fits into the system memory, CPU can process data at a much faster rate than the memory access speed which makes memory access a bottleneck.

- **Graphics processing unit (GPU)** [175]: is a specialized hardware designed to accelerate the creation of images in a frame buffer intended for display output.

  GPU has large number of processing cores as compared to a multicore CPU. In addition, GPU has its own high throughput DDR5 memory which is many times faster than a typical CPU's DDR3 memory. GPU performance has increased significantly in the past few years compared to that of CPU.

  Nvidia has released the CUDA framework [176] which made GPU programming accessible to all programmers without delving into the hardware details. This led to the development of faster machine learning algorithms such as GPUMiner [177].

  The primary drawback of GPUs is that once the data size is more than the size of the GPU memory, the performance decreases significantly as disk access becomes

the primary bottleneck. Another drawback is the limited amount of software and algorithms that are available for GPUs. Because of the way in which the task breakdown is required for GPUs, not many existing analytical algorithms are easily portable to GPUs.

- **Field programmable gate arrays (FPGA)** [178]: are highly specialized hardware units which are custom-built for specific sets of applications.

  FPGAs can be highly optimized for speed and can be orders of magnitude faster compared to other platforms for certain applications, but come with much higher cost. On the software side, coding has to be done in Hardware descriptive language (HDL) [179] with a low-level knowledge of the hardware, which increases the algorithm development cost. In recent years, the speed of multi-core processors is reaching closer to that of FPGAs.

3.2.2.5 Comparison of scaling techniques

Singh and Reddy authored an exhaustive survey on platforms for big data analytics [152]. This survey includes an extensive comparison of the scaling paradigms, based on the implementation of the K-Means algorithm using various platforms and computing models.

The comparison is summarized in the following tables: the first one compares the advantages and drawbacks of horizontal and vertical scaling in table 3.2, the second one compares the different platforms based on several characteristics using the (star) ratings, where 5 stars correspond to the best possible rating and 1 star corresponds to the lowest possible rating for any given platform for a particular characteristic.

Scaling	Advantages	Drawbacks
Horizontal	Increases performance in small steps as needed	Software has to handle all the data distribution and parallel processing complexities
	Financial investment to upgrade is relatively lower	Limited number of available software that can take advantage of horizontal scaling
	Can scale out the system as much as needed	
Vertical	Most of the software can easily take advantage of vertical scaling	Requires substantial financial investment
	Easy to manage and install hardware within a single machine	System has to be more powerful to handle future workloads and initially the additional performance in not fully utilized
		It is not possible to scale up vertically after a certain limit

Table 3.2 — Comparison of advantages and drawbacks of horizontal and vertical scaling.

Scaling	Paradigm	System/Platform			Application/Algorithm		
		Scalability	Data I/O performance	Fault tolerance	Real-time processing	Data size supported	Iterative task support
Horizontal	Peer-to-Peer (MPI)	★★★★★	★	★	★	★★★★★	★★
	MapReduce (Hadoop)	★★★★★	★★	★★★★★	★★	★★★★	★★
	MapReduce (Spark)	★★★★★	★★★	★★★★★	★★	★★★★	★★★
Vertical	HPC clusters (MPI)	★★★	★★★★	★★★★	★★★	★★★★	★★★★
	Multi-core CPU	★★	★★★★	★★★★	★★★	★★	★★★★
	GPU (CUDA)	★★	★★★★★	★★★★	★★★★★	★★	★★★★
	FPGA (HDL)	★	★★★★★	★★★★	★★★★★	★★	★★★★

Table 3.3 — Comparison of different platforms (along with their communication mechanisms) based on various characteristics.

The star ratings provided in Table 3.3 gives a bird's eye view of the capabilities and features of different platforms.

The decision to choose a particular platform for a certain application usually depends on the following important factors: data size, speed or throughput optimization and model development. Singh and Reddy [152] provide more details about each of these factors.

3.2.3 Application Layer: Analytics Packages

This layer is not mandatory for a well functioning big data analytics platform. However, it became essential thanks to its facilitating functionalities, since the computing models of the middle layer are low levels tools that are hard to learn and use. Therefore, some high-level parallel programming tools or languages are being developed based on these systems.

Indeed, there are several projects that provide higher levels of programming abstraction, through some advanced language systems, by building upon the computing layer in order to improve the efficiency of the big data analytics.

These include Google's *Sawzall* [180]; *Pig* [181], developed at Yahoo and included as part of Apache Hadoop; and Facebook's *Hive* [182], a data warehousing application that provides a SQL-like interface and relational model. All of these projects use MapReduce as the underlying computing model.

Similarly, Microsoft's *DryadLINQ* [183] programming system builds on Dryad and combines it with the .NET Language Integrated Query, which is used to integrate SQL-like language execution environment. Another example is *Berkeley data analytics stack (BDAS)* [184], an entire data processing stack based on Spark.

All these projects aims to help the programmers to design high-level analytics packages and tools, presented in the next section, used to easily provide high-level analytics

functionalities, like querying, reporting, scripting, machine learning and so on.

3.3 Big Data Analytics Tools & Frameworks

From the previous concepts and architectures, several big data analytics tools and frameworks have been developed.

Here the tools encompass software (libraries, packages and programming languages) used to design relatively small scale big data analytics processes, that not necessarily require tremendous computing power and storage capability. The frameworks are tools specifically dedicated to very large scale big data analytics on cloud platforms.

3.3.1 Tools

Many tools for big data analytics are available, including professional and amateur software, expensive commercial software, and free open source software [139][185]. In the following, the most used ones are presented.

- **R** [186]: one of the most well known open source programming language and software environment, is designed for data analysis and visualization. For computing-intensive tasks, code programmed with C, C++ and Fortran may be called in the R environment.

- **Weka** [187]: abbreviated from *Waikato Environment for Knowledge Analysis*, is another famous, free and open-source machine learning and data mining software written in Java. Weka provides such functions as data processing, feature selection, classification, regression, clustering, association rule, and visualization, etc.

- **Pentaho** [188]: is one of the most popular open-source BI software. It includes a web server platform and several tools to support reporting, analysis, charting, data integration, and data mining, etc., covering all aspects of BI. Weka's data processing algorithms are integrated into Pentaho and can be directly called.

- **Rapidminer** [189]: is an open source software used for data mining, machine learning, and predictive analysis, including Extract, Transform and Load (ETL), data pre-processing and visualization, modeling, evaluation, and deployment. RapidMiner is written in Java. It integrates the learner and evaluation method of Weka, and works with R. Functions of Rapidminer are implemented with connection of processes including various operators.

- **KNIME (Kostanz Information Miner)** [190]: is a user-friendly, intelligent, and open-source-rich data integration, data processing, data analysis, and data mining platform. It allows users to create data flows or data channels in a visualized manner, to selectively run some or all analytical procedures, and provides analytical results, models, and interactive views. KNIME was written in Java and is based on Eclipse EMF (Eclipse Modeling Framework).

KNIME controls data integration, cleaning, conversion, filtering, statistics, mining, and finally data visualization. Moreover, it is designed as a module-based and expandable framework. There is no dependence between its processing units and data containers, making it adaptive to distributed environment and independent development.

- **scikit-learn [191], mlPy [192]**: are open source, simple and efficient machine learning libraries in Python, used also for data mining and data analysis.

- **mlpack [193], SHOGUN [194]**: mlpack is a fast, flexible machine learning library, written in C++, that aims to provide fast, extensible implementations of cutting-edge machine learning algorithms. Shogun is and open source machine learning library that offers a wide range of efficient and unified machine learning methods, also written in C++.

3.3.2 Frameworks

Most of the available big data analytics frameworks rely on cloud computing and are divided into *Batch* processing frameworks and *Stream* processing frameworks

3.3.2.1 Batch Processing

Batch processing is a paradigm where a large volume of data is firstly stored and only then analyzed, as opposed to Stream processing. This paradigm is very common to perform large-scale recurring tasks in parallel like parsing, sorting or counting.

Most of the batch-oriented big data analytics are based on MapReduce, usually by means of Hadoop or Spark, like the ones in table 3.4.

Name	Specified Use	Advantages
Apache Mahout [195]	Machine learning algorithms in business	Good maturity, scalability, reliability
Radoop [196]	Big data analytics with Rapidminer and Hadoop	Ease-to-use, high scalability, reliability
Spark MLlib [197]	Scalable machine learning	Faster, ease-to-use, scalability
Jaspersoft BI Suite [198]	Business intelligence software	Cost-effective, self-service BI at scale
Pentaho Business Analytics [199]	Business analytics platform	Robustness, scalability, flexibility in knowledge discovery
Skytree Server [200]	Machine learning and advanced analytics	Process massive datasets accurately at high speeds
Tableau [201]	Data visualization, Business analytics	Faster, smart, fit, beautiful and ease of use dashboards
Talend Open Studio [202]	Data management and application integration	Easy-to-use, eclipse-based graphical environment

Table 3.4 — Big data analytics tools based on batch processing [45].

3.3.2.2 Stream Processing

Stream processing is a paradigm where data are continuously arriving in streams, at real-time, and are analyzed as soon as possible in order to derive approximate results. Due to its volume, only a portions of the streams are stored in memory [203]. It is used in online applications that need real-time precision.

The importance of stream-processing systems increases as more modern applications impose tighter time constraints on a particular event's propagation along the pipeline, leading to several tools (see table 3.5).

Name	Specified Use	Advantage
Storm [168]	Real-time computation system	Scalable, fault-tolerant, and easy to set up and operate
S4 [170]	Processing continuous unbounded streams of data	Proven, distributed, scalable, fault-tolerant, pluggable platform
SQLstream [204]	Sensor, M2M, and telematics applications	SQL-based, real-time streaming Big Data platform
Splunk [205]	Collect and harness machine data	Fast and easy to use, dynamic environments, scales from laptop to datacenter
Apache Kafka [206]	Distributed publish-subscribe messaging system	High-throughput stream of immutable activity data
SAP Hana [207]	Platform for real-time business	Fast in-memory computing and realtime analytic

Table 3.5 — Big data analytics tools based on stream processing [45].

3.3.3 Principles for designing Big Data systems

Big Data analytics are doomed to be more complicated than traditional data analysis systems. When trying to exploit Big Data, we not only need to develop new technologies, but also new ways of thinking in designing Big Data analytics systems.

In [45] the authors summarize seven necessary principles [208][209], given below, to guide the development of big data analytics solutions.

1. Good architectures and frameworks are top priority.
2. Support a variety of analytical methods.
3. No size fits all solutions.
4. Bring the analysis to data.
5. Processing must be distributable for in-memory computation.
6. Data storage must be distributable for in-memory storage.
7. Coordination is needed between processing and data units.

In summary, KDD as a workflow has known no drastic change. However, it has been technically boosted by means of data distribution and parallel computing, since they have a tremendous impact on reducing the overall process time and the bottlenecks. So, the KDD pipeline for big data is relatively the same, but each step evolved, thanks to distribution and parallelism, to handle big data leading to the big data analytics (see fig.3.2).

The development of cloud computing provides solutions for the storage and processing of big data. On the other hand, the emergence of big data also accelerates the development of cloud computing. The distributed storage technology based on cloud computing can effectively manage big data; the parallel computing capacity by virtue of cloud computing can improve the efficiency of acquisition and analyzing big data.

In this scope, numerous solutions have appeared, each one bringing unique functionalities and characteristics. Therefore, the main concern of the big data analytics users have shifted from "how to design from scratch a good analytics process?" to "Which available solutions is the best suited and how to tune it for better results?". Nonetheless, big data analytics is a work-in-progress that continuously needs to evolve in order to handle bigger data, with evolving and rising challenges, especially in the context of complex systems .

4 Data Science Challenges

At the beginning of the big data era, the first data scientists struggled against barriers risen from the very nature of these new data. The analytical tools, mainly statistical and mathematical methods, as well as storage and computing platforms available at that time, were not suited to cope with such big data.

So, in order to express their urge for new tools able to handle big data, in relation to the hardships of their analysis process, big data 'challenges' have been introduced. These first challenges are technical ones: they embody the features and characteristics that those who deal with big data analytics must keep in mind during the design process, to ensure the production of adequate analytics tools. With time, many challenges have been defined to cover all aspects of data science, not only the technical ones, like the management of the data (privacy and veracity) and the relevance of the analysis (timeliness and value).

In this chapter, we introduce the data science challenges that are most common and accepted by the data science community. The relentless efforts to overcome these challenges led to data science competitions, in order to give birth to original and/or more efficient big data analytics tools. In addition, we define new challenges in relation to the new evolution of the data world, which is caused mainly by the explosion of data from fast-generative, mobile and pervasive devices.

4.1 Ongoing Challenges

4.1.1 Big Data '3Vs'

Most definitions of big data focus on the size of data in storage. Size matters, but there are other important attributes of big data, namely data variety and data velocity. The three 'Vs' of big data (volume, variety, and velocity) constitute a comprehensive definition, and they bust the myth that big data is only about data volume. In addition, each of the '3Vs' has its own ramifications for analytics (fig.4.1).

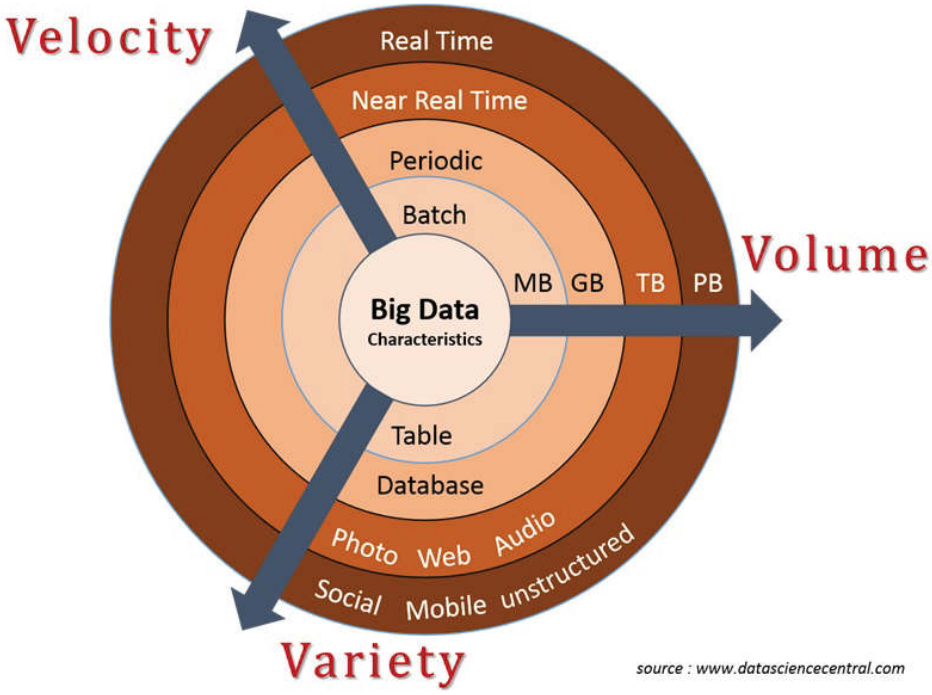

Figure 4.1 — The big data '3Vs' model [210].

The initial '3Vs' model, three main challenges inherent to the characteristics of big data, was introduced by Doug Laney -an analyst of META (presently Gartner)- in a research report [8]. Then, this '3Vs' model has been used and spread all over the literature and became a common concept. We summarize these challenges as follows:

- **Volume:** data sets with tremendous size (lines, entries, records) and high complexity (a lot of features, attributes, variables). It's obvious that data volume is the primary attribute of big data. With that in mind, most people define big data in terabytes and sometimes in petabytes.
- **Velocity:** rapid generation of data that arrives in batch, near real-time, real-time or continuous streams. This velocity begets new network and storage architectures, for the sake of instant access without loss.
- **Variety:** different types of data come in various forms. One of the things that makes big data really big is that it's coming from a greater variety of sources than ever before.

Many of the newer ones are Web sources, including logs, click-streams, and social media. Sure, user organisations have been collecting Web data for years.

But the recent tapping of these sources for analytics means that the so-called structured data (which previously held unchallenged hegemony in analytics) is now joined by unstructured data (text and human language) and semi-structured data (XML, RSS feeds). There's also data that's hard to categorize, as it comes from audio, video, and other devices. Plus, multidimensional data can be drawn from a data warehouse to add historic context to big data.

That's a far more eclectic mix of data types than analytics has ever seen. So, with big data, variety is just as big as volume. In addition, variety and volume tend to fuel each other.

In the past, these challenges, also known as the 'data flood', were mitigated by processors getting faster, following Moore's Law. But there is a fundamental shift under way now: data volume is increasing faster than CPU speeds and other computer resources. Due to power constraints, clock speeds have largely stalled and processors are being built with increasing numbers of cores.

In short, one has to deal with parallelism within a single node. Unfortunately, parallel data processing techniques that were applied in the past for processing data across nodes do not directly apply for intranode parallelism[1], since the architecture looks very different. Consequently, storage systems and processing techniques at that time were driven to their limits.

4.1.2 Big Data '5Vs'

After becoming familiarized with the first challenges, new techniques started to get good results, but soon the data flood overwhelmed these techniques.

Indeed, as the volume of data grew and the sources multiplied, raw data were becoming poorer and useful information becoming rarer (*"thirst despite the flood"*). Usefulness and reliability of the data and their sources were increasingly questioned. Hence the apparition of two new challenges bringing the challenges of big data to '5Vs'.

Jagadish et al. [105] define the new 'Vs' as follows:

- **Value:** the usefulness of the data or more precisely the amount of useful information among the data flood, which may give a full play to the economic function, improve the productivity and competitiveness of enterprises and public sectors, and create huge benefits for consumers. The economical and human benefits of big data for the US were reported by [30].
- **Veracity**: reliability and confidence attributed to the data and their sources. As a result of the age of automation, most of the data gathering implies and relies on the use of machines, mainly computers to store information but also sensors and other devices that record the state of a the cyberphysical world.

[1] We need new tasks scheduling politics to use multi-cores CPUs efficiently and fairly

Despite their growing reliability and efficiency, machines can make mistakes because of a failure or their misuse and calibration by the humans, intentionally or not. These mistakes are translated as noise (errors, missing values...) in the data. As a result, taking into account the veracity of the data is important to get noise-resilient big data analytics tools.

4.1.3 Other challenges

Since companies, institutions, organisations are now using or planning to use big data analytics to boost their productivity and offer new products and services, a new challenge related to the minimization of the reactivity of the tools or response time has emerged.

Unfortunately, to achieve their goals, these organisations rely mostly on people's (customers, users, citizens...) personal data, which led to a growing fear of who can see their personal data and how they are used. The people's concern of their privacy and digital safety is another challenge which needs to be overcome to keep at bay their misuse.

- **Privacy & Security:** A lot of big data contains personal information about individuals or organisations. People are concerned about how information relating to them is used and by whom, particularly if their data are used to affect them. The purpose behind this challenge is developing trackable privacy-preserving data analytics tools that would allow to use the data while keeping them anonymous.

  For example, recently the European Union established the General Data Protection Regulation (GDPR) [211], a set of stronger rules on data protection which means people have more control over their personal data and businesses benefit from a leveled playing field. The GDPR is the most important change in data privacy regulation in 20 years.

- **Timeliness:** since computing power is growing, chips are shrinking and networks capabilities are increasing; the generation and the acquisition rates of data is way beyond the storing rate and the processing rate of the current systems, as demonstrated in fig.4.2 ([212]). This is especially true when unpredictable burst-type data generation occurs. Consequently, we need tools able to analyze data in real-time in order to, on one hand, reduce the computing power used to process all the data and, on the other hand, avoid to store all the data.

4.2 Data Science Challenges in Big Data Analytics

The ongoing data science challenges can be seen as high level problematics that can be translated to the technical level. To do so, we take the previously (chapter 2) defined steps of the big data analytics pipeline and describe how the data science challenges are present in each step.

- **Data acquisition:** big data are records of some underlying activity of interest. Much of this data can be filtered and compressed without any loss of usefulness. One challenge

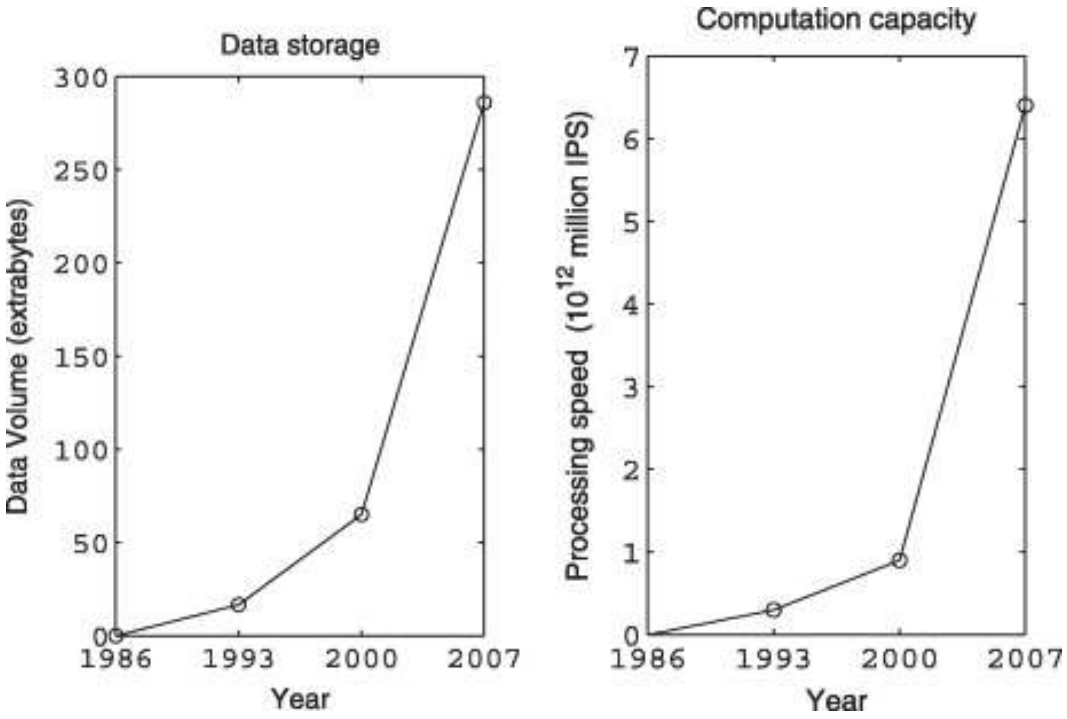

Figure 4.2 — Data flood: the growth of data volume has surpassed the computational capabilities [212].

is to define these "on-line" filters to do so. Furthermore, loading of large data sets is often a challenge and we need to push summarization to edge devices, possibly combined with efficient incremental ingestion techniques[2].

- **Information extraction and cleaning:** Information will usually not be in a format ready for analysis. It is thus necessary to have a relevant extraction process that pulls out the required information. Such extraction is often highly application-dependent and most data sources are notoriously unreliable. Understanding and modeling these sources of error is a first step toward developing data cleaning techniques.

- **Data integration, aggregation, and representation:** effective large-scale analysis often requires the collection of heterogeneous data from multiple sources. This heterogeneity resolution leads to integrated data that is uniformly interpretable within a community. The cost of full integration is often formidable. Moreover, the massive availability of data on the Internet, coupled with integration and analysis tools leads to problem of tracking the provenance.

- **Modeling and analysis:** querying and mining big data are fundamentally different from traditional statistical analysis on small samples. Big data is often noisy, dynamic, heterogeneous, inter-related, and untrustworthy.

  Nevertheless, even noisy big data is generally more valuable than insufficiently sized samples. General statistics usually overpower individual fluctuations and often

[2]Techniques that reduce the size (lines and/or dimensions) of the data, like compression algorithms

disclose more reliable hidden patterns and knowledge. So, one can use approximate analysis to get good results without being overwhelmed by the volume.

- **Interpretation:** involves examining all the assumptions that were made and retracing the analysis. This step provides users with the ability both to interpret results and to repeat the analysis with different assumptions.

  The result of interpretation is often the formulation of opinions that annotate the base data. It is common that such opinions may conflict with each other or may be poorly substantiated by the underlying data, when many users rely strongly on their presumptions or beliefs about the data and their domain before the analysis.

4.3 Data Science Competitions

The importance of overcoming the data science challenges more efficiently, gave birth to several international competitions, in order to promote research works, compare them, evaluate them, improve them, allow exchange and cooperation between research teams and match the academic world with the industrial one to share their efforts in this matter.

Here is a non-exhaustive list of the most interesting data science competitions:

- **Knowledge Discovery from Data Cup (KDD Cup)** [213]: perhaps the most well known competition of data mining and knowledge discovery fields, organised by the ACM (Association for Computing Machinery) special interest group on knowledge discovery and data mining, the leading professional organisation of data miners. Usually, multiple submissions (maximum of 5 entries per day) are allowed until the submission deadline, from individual or team competitors.

  The submitted results are evaluated on *Area Under the ROC Curve (AUC)* [214], a metric which compares the expected outcomes and the competitors observed outcomes. The KDD Cup topics are versatile, ranging from profit optimization, clickstream analysis, particle physics, paper publication, breast cancer and pulmonary embolisms detection, for instance.

- **International Joint Conferences on Artificial Intelligence (IJCAI)** competitions: each year IJCAI releases several artificial intelligence competitions. Since artificial intelligence has a big part in data science, some of these competitions are dedicated to big data analytics.

  For example in 2018 one competition -the "Alimama International Advertising Algorithm" [215] competition- objective was improving, by designing and applying new machine learning algorithms, the accuracy of the probability of a user to purchase the displayed item after clicking this item, from massive users transactions data of a famous online retailer.

- **European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) challenges** [216]: is one of the leading academic conferences on machine learning and knowledge discovery, held,

in Europe every year. Like IJCAI, ECML/PKDD organises challenges related to the conference topic. For example in 2015, one challenge was to predict the destination of taxi trips based on initial partial trajectories in Porto, Portugal.

- **TEXATA championship** [217]: is a fun, innovative and challenging annual big data analytics world championship for students and professionals to develop and test their big data, data science and business analytics skills against friends, colleagues and top data scientists around the world.

  TEXATA mission is to help build practical business skills and educational awareness of the future importance of analytical skills in Science, Technology, Engineering and Mathematics (STEM). TEXATA is focused on big data impacts for real-world enterprise and business decision-making. Participants address data sets across three core sectors: Financial Services, Mobile Data and Social Data.

In spite of the breeding grounds conducive to innovation provided by such competitions, the new big data analytics tools are more and more sophisticated, requiring advanced knowledge about their domain target and extensive programming skills, which make their adaptation and update more difficult and their need for storage and computing power greater. Therefore, the necessity of new flexible techniques arise.

4.4 Rising Challenges

Many of the current analytics tools [218] [219] can handle ongoing challenges and provide satisfactory results with reasonable cost. For example the recent *Data Science Machine* [71] is based on a *Deep Feature Synthesis* algorithm and an automatically optimized ensemble learning[3] pipeline in the interest of providing a fully automated analytics tool.

However, with the recent increase of the number of smart and wearable devices and other measuring instruments in ambient applications, we barely begin to deal with *all* aspects of those new big data. As a result, the importance of the ongoing challenges is renewed, and additional complementary data processing and data management challenges appear.

As [220] expected, we move forward to the next stage of big data analytics. We need to define new data science challenges in response to the actual and future evolution of the digital world. Thence, I propose the following new challenges:

- **Genericity:** as shown in [218] and [219], most of the analytics tools are domain-dependent and require domain-specific expertise to build these tools. Hence, in order to adapt these tools to other application domains, the designer has to reconstruct his processing technique (data mining algorithm).

  Thus, designing a generic big data analytics tool should be a new challenge, which aims to lower and ultimately to get rid of the preconceived hypotheses and domain expertise that we need about the data in order to choose and configure the right analytical tool for running the data analysis.

[3] Combination of classification algorithms.

- **Elasticity:** with the spread of grid, clouds, fogs and other kind networked machine with computational capabilities, the overall computing power is growing which benefits the analytics tools, especially with the available frameworks that ease the use of computing clusters. However, some knowledge of these frameworks is required in order to adapt the big data analytics to the available computing clusters.

  For this reason, we define the need of flexible big data analytics as a challenge, that should result in self-adaptive tools that dynamically exploit the available computing power. In other words, provide big data analytics tools able to grow or spread when the computing power grows, when new machines or processors are added to the current computing environment, and shrink when the available computing power diminishes.

- **Scalability:** when dealing with open systems, like *IoT*, the umber of new data sources (devices) that might be incorporated into the system is unpredictable. Therefore, the analytics tools dealing with such dynamic environments should be able to cope with their explosive growth and keep working after any expansion, while preventing a huge deterioration of the performances of the analytics process or a breakdown.

  This new challenging feature may seem analogous to the elasticity, defined previously, but the latter is one way to achieve the former. Thus, scalability can be reached by increasing the required computing power (elasticity challenge) as little as possible, linearly in the worst acceptable case and logarithmically in the best case, by updating the current resources use strategy to focus on the most important tasks, or by combination of the elasticity and the exploration strategy update.

- **User Feedbacks Integration:** human knowledge or intuition can be useful to speed up the analytics process and increase its efficiency. However, most of the conventional analytics tools provide little or no means for an efficient integration of user feedbacks. Usually they allow the user to incorporate its feedbacks, by means of annotations for instance, once the analytics process ends and restart the process with the same or new data.

  This can be tedious and unproductive due to the time loss caused by the postponed consideration of the user feedbacks. As a consequence, we define as a new data science challenge the ability of an analytics tool to take in user feedbacks and update the analytics process accordingly on the fly. In other words, during the data processing the user can guide the analytics process by indicating which hypotheses or data space regions he deems relevant to focus on or to avoid.

- **Variability:** in addition to the changes (growth and evolution) to data content, the data structure (features) may change as well over time. But, the current big data analytics are rigid pipelines, due to the immutable data structures. So, we express this challenge as the ability to handle data structure changes on the fly, by means of dynamic reconfiguration or reconstruction the analytical process/algorithm to adapt the analysis to the new data.

  Therefore, we consider that managing the dynamicity of the data (content and **structure**) must be acknowledged as a new crucial challenge.

- **Visibility:** we define the visibility as the issue of finding drowned information in data and retrieve regularities as well as irregularities and relations that can only be seen through a time-line.

As the big data community shows a real interest in handling dynamic data (evolving content), new data mining techniques known as *stream mining* were developed [221] [222]. These stream mining algorithms usually sample the data stream (pick up some points) in a certain manner and process them in an incremental or on-line way.

Despite the outcomes delivered by these new techniques, the data stream is under-exploited which potentially leads to a leak of useful information on one hand and forgetting what was previously discovered on the other hand.

Thence, our aim is to design new big data analytics techniques that can manage truly dynamic big data (content and structure), in a domain-agnostic way, and adapt itself to the changes that occur over time without having to shut down the data processing to take into account these changes by updating the process and restarting all over again.

AMAS4BigData
Adaptive Multi-Agent Systems
for Dynamic Big Data Analytics

The AMAS4BigData framework

Abstract

THE big data era brought us new data processing and data management challenges to face. Existing state-of-the-art analytics tools are now able to handle ongoing challenges and provide satisfactory results with reasonable cost. However, the speed at which new data is generated and the need to manage changes in data both for content and structure lead to new rising challenges. This is especially true in the context of complex systems with strong dynamics, as in for instance large scale ambient systems with network of sensors (*IoT*).

This second part presents and describes a response, to these new challenges, as a new generic, flexible, and elastic big data analytics framework, called *"AMAS4BigData"*, based on a collective bio-inspired artificial intelligence that handles well dynamic complex systems.

Chapter 5, presents the origins and an overview of the new framework, relying on existing elements in the literature. Chapter 6 goes further by giving an extensive description of the theoretical inspirations and architecture. Finally, chapter 7 shows the fulfillment of this framework, by means of its implementation in a dynamic big data analytics system *"DREAM"*.

5 Dynamic Big Data Analytics or Complex System Analytics

After introducing data science and its main concept, big data analytics, as well as the related challenges, I introduce in this chapter a new big data analytics paradigm and architecture, *dynamic big data analytics*, which I consider as the future of the field. To come up with this new analytics model, I rely on artificial intelligence concepts and technologies that are suited for handling the problematic of big data processing under an original point of view, that of data produced by complex systems.

5.1 Future of Big Data Analytics: *Dynamic* Big Data Analytics

As discussed in chapter 4, from the design of data science tools, and more specifically big data analytics, arise several challenges related to the current and future limitations of the conventional big data analytics.

To overcome these challenges, data scientists must consider and fulfill the following requirements in order to build the future big data analytics, which I designate as *Dynamic Big Data Analytics*.

- **Unified Analytics Architecture**: it is not clear yet how an optimal architecture of an analytics system should be constructed to deal with big data. Nevertheless, such architecture has to insure, for the system, fundamental properties: scalable, general,

extensible, robust and fault tolerant, minimal maintenance, and debuggable.

- **Handle Time Evolving Data**: big data are more often streams evolving over time, so it is important that big data analytics should be able to adapt and detect changes. For example, the stream big data analytics field has very powerful techniques for this task [223].

- **Real-Time Analytics**: data may be impossible to store, because storage is not allowed or the data production rate is bigger than the rate of the storage system, and/or the analytics throughput may be critical. In that case, what is important is not the size of the data, but their *recency* or *freshness*. In other words, when new data come they must be processed as fast as possible.

- **Dynamic Visualization**: an important task of big data analytics is how to visualize the results. As the data is so big and evolving, it is very difficult to find user-friendly visualizations. New techniques, and frameworks to show the evolution of the extracted knowledge will be needed.

- **Retrieve and Expose Drowned Information**: traditional data mining techniques tend to, roughly speaking, summarize and compact the data. Thus, some peculiar information may be statistically drowned, especially when trying to discard noise.

  For instance, at first the Higgs boson[1] was rejected as noise. But after a second pass, the existence of the particle was established thanks to the use of many statistical techniques and classifiers dedicated to study only the signal of this particle [224]. So, The Higgs boson discovery provides an example for the data mining methods maximizing signal significance over background noise models.

- **Distributed Mining**: many data mining techniques are not easy to distribute, meaning to break them into sub-tasks and parallelize[2] them. Despite its difficulty, having a distributed analytics system is a major advantage, if not essential, when dealing with tremendous quantities of heterogeneous data coming from various sources. Moreover, distribution allows to efficiently use horizontal computing models (3.2.2.3).

- **Non-Centralized Analytics**: to keep up with the *Data Flood*, conventional big data data analytics systems require more storage capabilities and computing power, by reason of their centralized processing pipeline.

  Moreover, even if distributed, analytics systems have ultimately one central unit, or very few ones, that gathers the processing sub-results to compute the final result. This centralization induces a bottleneck equivalent to a cognitive overload in a biological brain when the number of inputs or perceptions reaches the limit of the analytics system.

  Therefore, the central unit of an analytics system, responsible of the information aggregation, the control and decision, has to be fragmented/distributed in many units that interact with each other to collectively build the final result. Put it simply, the knowledge must be non-centralized (distributed).

[1] An elementary particle in the Standard Model of particle physics.
[2] Making an algorithm able to use parallelism.

5.2 Dynamic Big Data Analytics Paradigm

In order to satisfy the previously defined requirements of dynamic big data analytics, one must rely on a new paradigm. In this section, I present this paradigm through its origins.

5.2.1 Multi-Agent System paradigm: Agent Mining

Agents[3] comprise a powerful technology for the analysis, design and implementation of autonomous intelligent systems, known as *Multi-Agent Systems (MASs)* [225], that can handle distributed problem-solving, cooperation, coordination, communication, and organisation in a multiplayer environment. Multi-agent systems and the data mining (KDD) approaches have merged leading to *agent mining* field [226].

Several examples of agent mining systems, like [227][228][229][230], were designed. Some are dedicated to one task, others propose a more general framework to achieve several mining tasks. The set of these systems brings the following advantages:

- The synergy of agents and data mining[4].
- Efficient high level knowledge managing;
- Extendibility of the analytics frameworks;
- Resource and experience sharing;
- Greater end-user accessibility;
- Information hiding and addressing privacy and security issues.

There is an increasing interest in MAS technologies and their applications on big data analytics [231][232]. Several researchers try to use concepts like swarm intelligence, self-organising maps, etc. However, all these new techniques are still domain-dependent and do not handle changes in the data.

So, even if the field of agent mining has grown during the last years, the available systems are still not able to provide dynamic big data analytics, as a result of their inability to fulfill all the characteristics of dynamics big data analytics as defined in the first section.

Nonetheless, agent mining based systems show some improvements compared to conventional big data analytics. Therefore, the Multi-Agent System (MAS) approaches are a good track for implementing dynamic big data analytics and the limitations of the MAS paradigm is inherent to pipeline-like architecture of big data analytics.

Thus, instead of trying to improve big data analytics while preserving it architecture, we better have to change the architecture by changing our point of view on the big data.

[3] Autonomous entities (sub-programs), that have some knowledge and skills for achieving their tasks.

[4] Agents can be used to mine large data collections efficiently. Knowledge models extracted via data mining can make agents more efficient.

5.2.2 Digital Universe: Complex Systems Analytics

Complexity of the near future and even nowadays digital universe[5] is exponentially increasing, resulting in the data flood. The reasons are the presence of non linear interactions, the facts that knowledge and control have to be distributed, that the system is more and more often open, its environment dynamic and the interactions unpredictable.

Thus, to process the data of this complex digital universe, the analytics system has to be *self-adaptive* like a natural system (bird flocking, animal herding, bacteria...) which is non-centralized and thrives even in harmful and highly dynamic environments.

Designers of complex systems have been taking inspiration from natural systems in which complex structures or behaviours *emerge* at the global level of a system from interactions among its lower-level components.

Different kinds of techniques have been developed such as those based on heuristics and metaheuristics [233], those based on learning such as genetic algorithm or neural network [234], [235] and those based on self-organisation processes [236], [237], which has led to the family of bio-inspired collective artificial intelligence.

One such artificial intelligence is the Self-Adaptive Multi-Agent System paradigm (fig.5.1) presented in the next chapter.

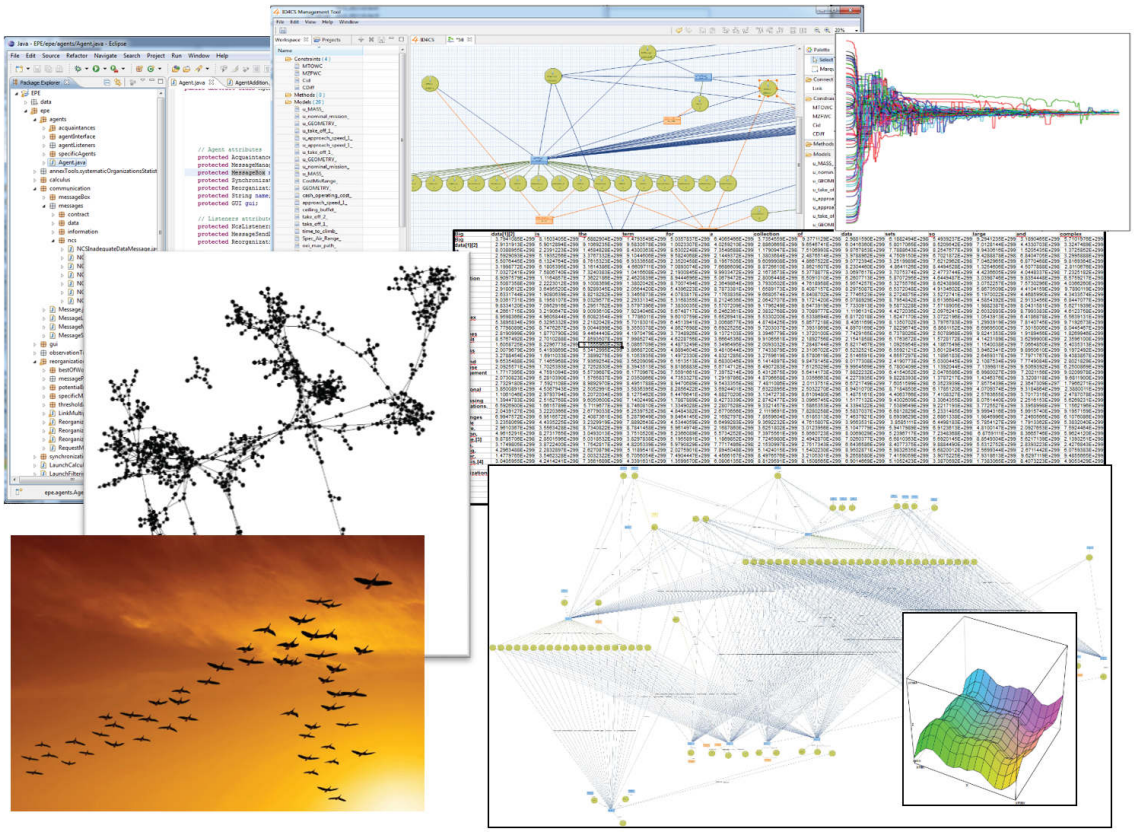

Figure 5.1 — Self-Adaptive Multi-Agent Systems: from the inspiration from natural systems to artificial systems tackling complex and decentralized tasks.

[5] The collection of the data generative applications and devices.

5.2.3 Self-Adaptive MAS Paradigm for Dynamic Big Data Analytics

Since a Multi-Agent System (MAS) is defined as a macro-system composed of autonomous agents which pursue individual objectives and which interact in a common environment to solve a common task, it can be viewed, when extended with mechanism of self-adaptation through cooperation, as a paradigm to handle dynamics big data originated from the complex digital universe.

Conventional big data analytics processes (fig.5.2(a)), which are rigid straightforward pipelines, do not allow the modification on the fly of the content and most importantly the **structure**) of the data already loaded in the analytics process. Moreover, despite the attempts to bypass this rigidity through processing time reduction by means of distribution (fig.5.2(b)), for example with the help of the *MapReduce* computing model, the main issue about the dynamic update still remains.

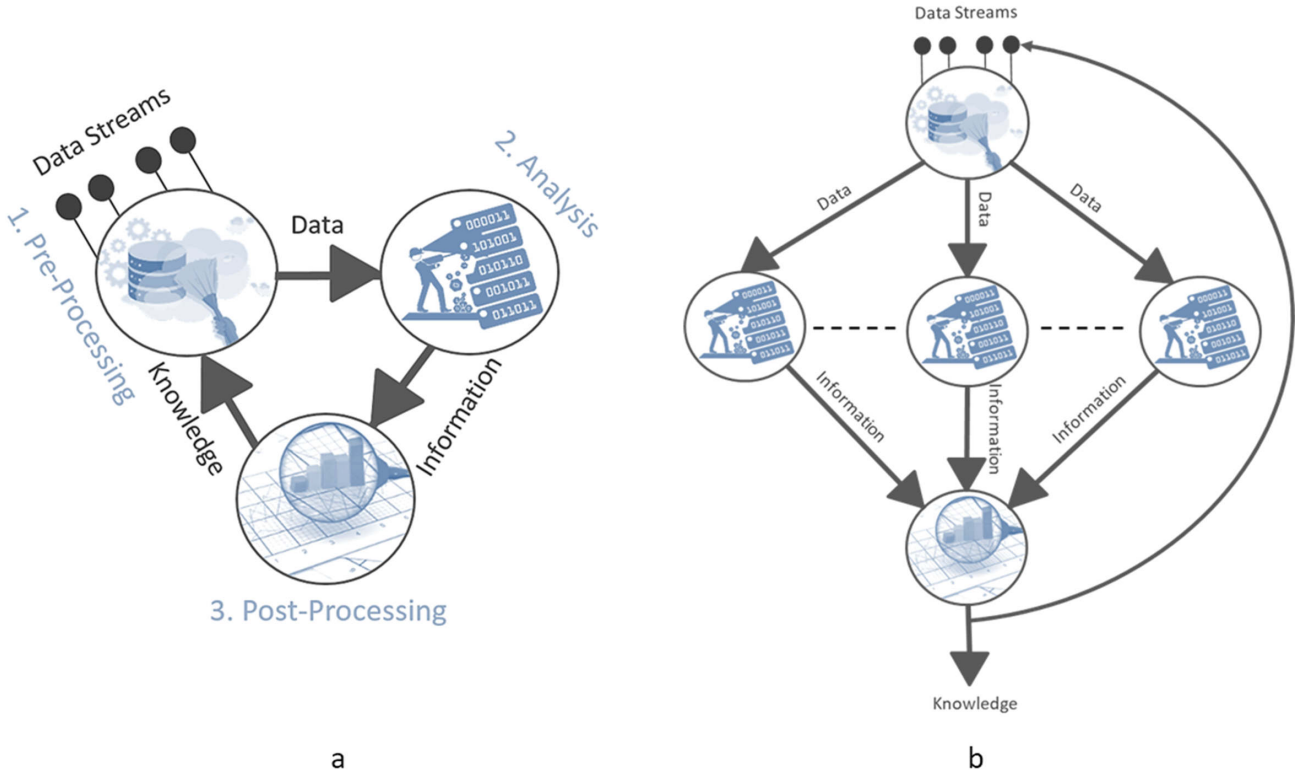

Figure 5.2 — Conventional Big Data Analytics Architecture: (a) high/macro level, (b) low/micro level.

The self-Adaptive Multi-Agent System (AMAS) technology, with the cooperative interaction process of its autonomous agents, gives us the means to break down the rigidity of conventional big data analytics and *decentralize* them (see fig.5.3).

This decentralization is translated by the split of the big data analytic process into sub-processes, each one dedicated to one input data stream. The analytic sub-processes interact, through communication, so they can help each other and work together for the sake of a continuous real-time adaptation of the global analytic process to data changes, resulting in full-fledged dynamic big data analytics.

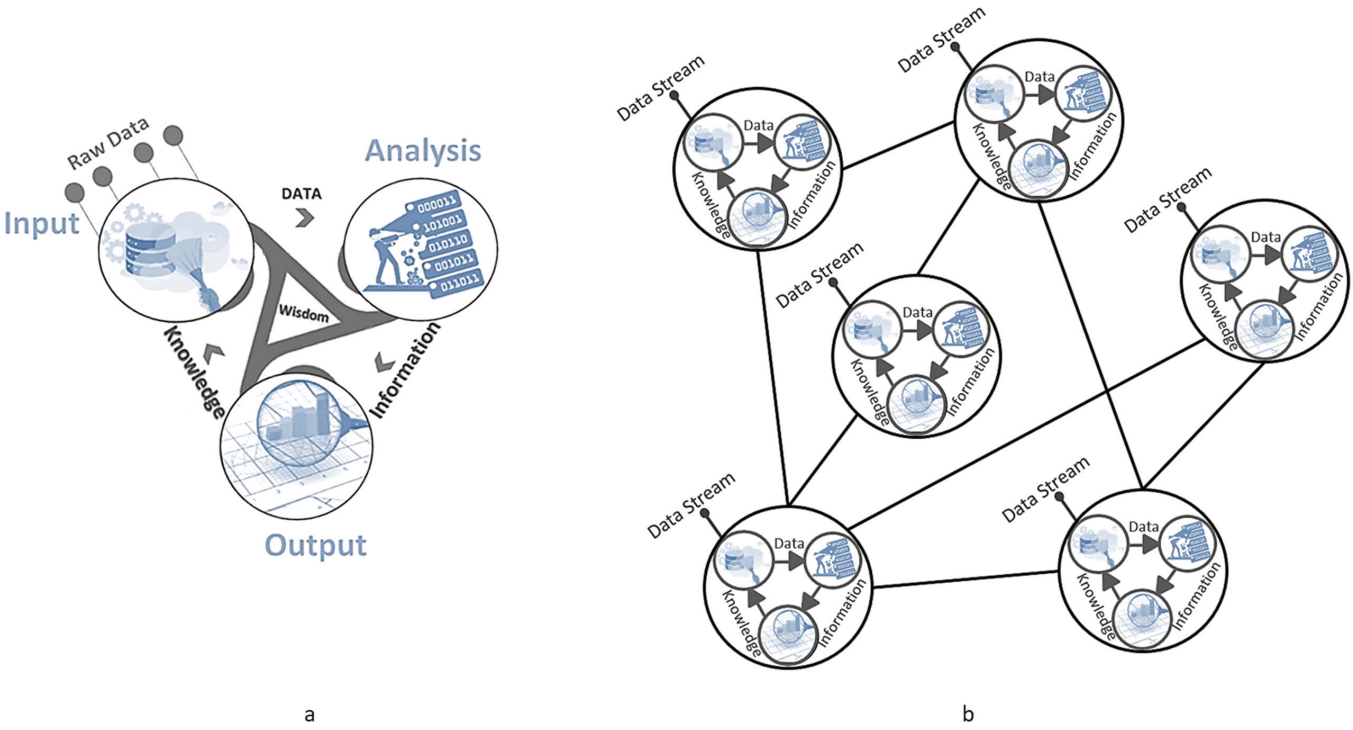

Figure 5.3 — Dynamic Big Data Analytics Architecture based on AMAS: (a) high/macro level, (b) low/micro level.

The detailed description of the AMAS based dynamic big data analytics framework is the object of the next chapters.

5.3 Definition of Dynamic Big Data Analytics

I sum up the definition of dynamics big data analytics by means of three concepts and three dimensions. This definition may not always be the best suited or the most common, but it holds the main keys that one should keep in mind when it comes to dynamic big data analytics, and gives a global picture with enough details in order to have a fine overview of it (see fig.5.4).

5.3.1 Concepts of Dynamic Big Data Analytics

- **Data.** Obviously the core of dynamic big data analytics. Data consists here in a huge amount of potentially private and critical data generated rapidly from multiple sources in various shapes and evolving over time. Data may be volatile, non persistent and untrustworthy due to the possible uncertainties within them. Furthermore, new data sources can appear and others disappear.

- **Processing.** Knowledge Discovery from Data (KDD) workflow that relies on the newly defined paradigm of dynamic big data analytics (5.2), so as to satisfy the requirements of future big data analytics (cf. section 5.1).

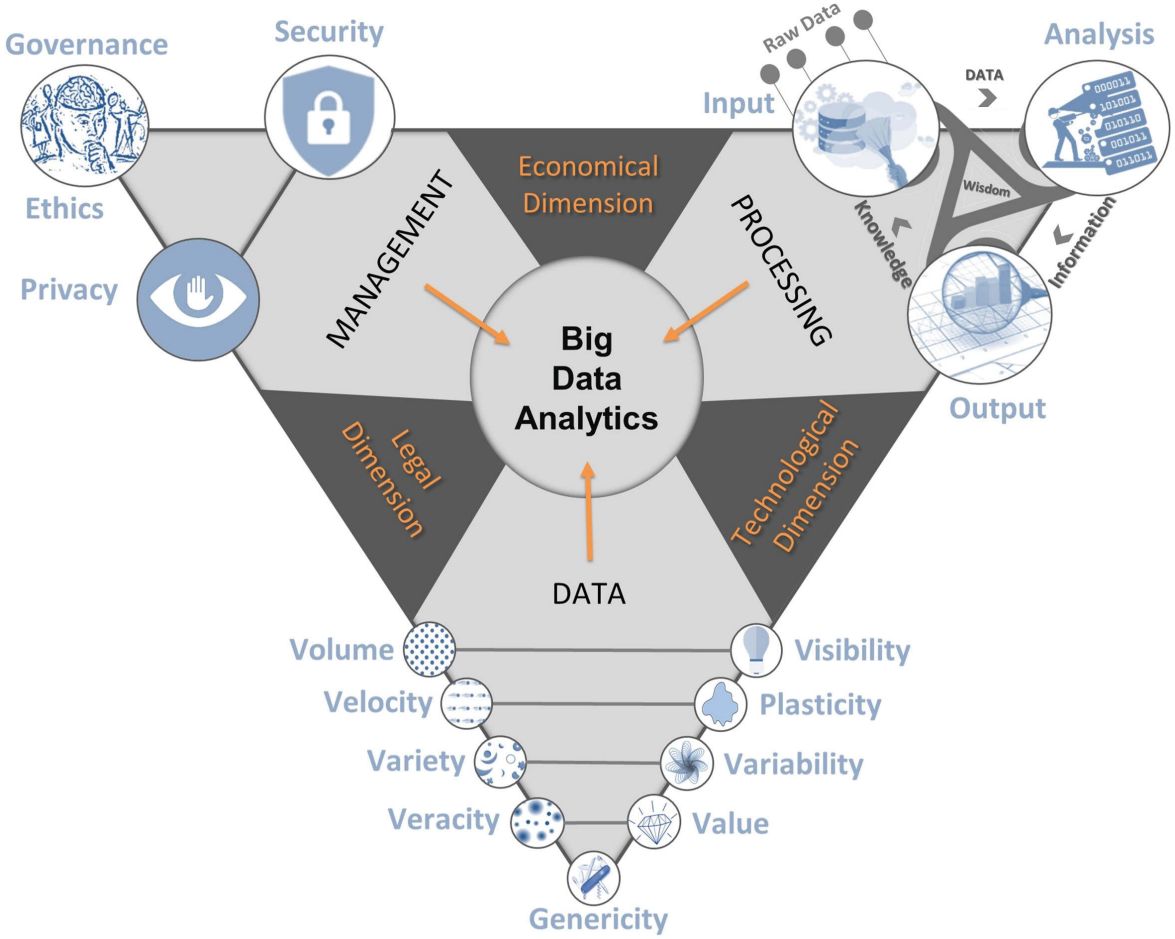

Figure 5.4 — An overview of Dynamic Big Data Analytics.

▷ **Management.** In the past few year, people's concern about malicious or unethical use of their personal and private data or critical ones, like the human genome and medical data [238], in favor of making money and other benefits, has grown.

This led the big data analytics designers to set rules for safe data management that ensures *privacy*, *security* and *ethical governance* of such personal and sensitive data. For instance, a widespread technique to achieve this is to encrypt the *Personal Identifiers (PID)*, like the name, the birth date and the zip code, during the integration task in the pre-processing step [239] and remove them for the analysis step.

5.3.2 Dimensions of Dynamic Big Data Analytics

Each dimension of big data analytics relies on two concepts.

▷ **Technological dimension.** Or IT-infrastructure dimension; relies on the *data* concept and the *processing* concept. Thus, this dimension represents the set of tools, software and hardware architectures used for data storage and data processing, like for example: *DBMS*, *MapReduce*, *Grid Computing*, *Cloud Computing* and so on.

- **Economical dimension.** Joins the *management* concept and the *processing* concept, and embodies the commitment of the data analysts to discover new processing methods that get more relevant and less costly information in order to earn as much benefits as possible in a safe management scope.

  One could understand 'benefits' as making profit like in a business intelligence sense [240], but it is not necessarily the case. Actually, the economical benefit can be seen as 'the greater good' at the societal level (health-care [241], smart cities [242][243], etc.)

- **Legal dimension.** Depends on *management* and *data*. It expresses the obligation of ethical and safe use of the data, for example by means of giving to the owner the full rights and control on his data and also giving him a clear description of how his data are used, for what purpose and by whom. Many organisations and councils were founded to establish and enforce laws in this scope.

To conclude, a comprehensive approach aiming at producing generic and powerful dynamic big data analytics, has to cover a broad range of these challenging elements. This must be done by changing the conventional point of view on big data and advocate new suitable paradigms. I define such an approach in the next chapter.

6 AMAS4BigData Framework Description

Let us now dive into the description of the dynamic big data analytics framework that I propose, *AMAS4BigData*, through the formalization of its core concept, the *self-Adaptive Multi-Agent System (AMAS)* theory, and the description of this theory implementation which gives the framework its architecture.

6.1 AMAS4BigData Core Concept: AMAS Theory

Self-Adaptive Multi-Agent Systems (AMAS) are Multi-Agent Systems (MAS) with special features. So first, the definition of Multi-Agent Systems is given. Then, we will add the specificities of AMAS.

6.1.1 Multi-Agent Systems

Definition 1. *A multi-agent system (MAS) [225] is a distributed system composed of several autonomous software entities (the agents), interacting among each others (usually by sending information and request messages) and with their environment (by observing and modifying it) as illustrated in fig.6.1 . The* autonomy *of an agent is the fundamental characteristic that differentiates it from, for example, the computer science concept of object.*

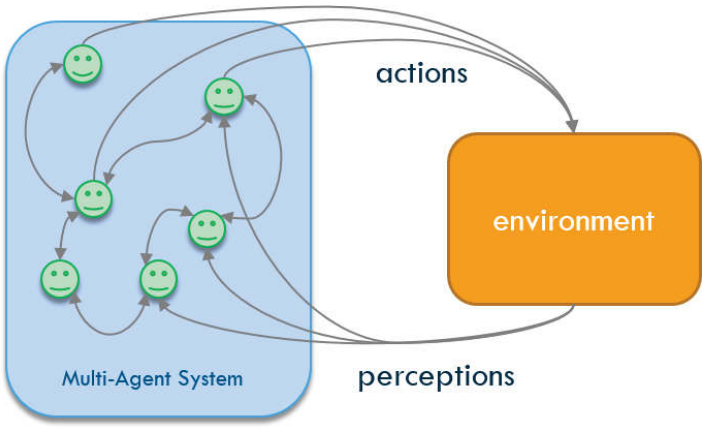

Figure 6.1 — Multi-Agent Systems overview (Nicolas Verstaevel PhD defence [244]).

While an object is a passive entity encapsulating some data and functions, waiting to be solicited, an agent on the other hand is capable of reacting to its environment and displaying pro-activity (activity originating from its own decision).

From this comparison it should be clear that the concept of agent is, like the concept of object, the building brick of a paradigm which can be used to model a complex reality in a bottom-up way, relying only on a limited and localized knowledge of the environment for each agent. And indeed, agents have been used in a great variety of fields, a fact which can contribute to explain the difficulty to produce a unified definition of the concept.

While it is not true for all MAS, some interesting properties can be achieved when taking advantage of the autonomy of the agents. This autonomy, coupled with an adequate behavior of the agents, can lead to systems able to adjust, organise, react to changes, *etc.* without the need for an external authority to guide them.

These properties are gathered under the term *self-** capabilities [245] (self-*tuning*, self-*organising*, self-*healing*, self-*evolving*...).

Not all MAS necessarily present all of these self-* capabilities but, as a result of building a system from autonomous and locally situated agents, many MAS will exhibit them to some

degree. Consequently, MAS are often relevant for dynamically taking into account changes in their environment. For example, a MAS in charge of regulating the traffic in a city should be able to react efficiently to disturbances, like accidents or traffic jams.

MAS have been applied to a great variety of fields: social simulation, biological modelling, systems control, robotics, *etc.* and agent-oriented modelling can be seen as a programming paradigm in general, facilitating the representation of a problem.

6.1.2 Self-Adaptive Multi-Agent Systems

Definition 2. *A self-Adaptive Multi-Agent System (AMAS) [246], [247] is a MAS relying strongly on self-* properties. Thus it is able to adjust itself, organise itself, heal itself,* etc. *to remain in a well-functioning state after a perturbation.*

A designer following this approach focuses on giving the agent a local view of its environment, means to detect problematic situations and guidelines to act in a *cooperative* way, meaning that the agents will try to achieve their goals while respecting and helping the other agents around them as best as they can [246].

The fact that the agents do not follow a global directive towards the solving of the problem but collectively build this solving, produces an *emergent* problem solving process that explores the search space of the problem in original ways.

The difficulty here is to give the agents the right behavior in order to get the right global function and a good adaptation capability since there is no formal process, which translates the behavior of the components and their interactions into a well-defined global function.

Definition 3. *A behavior of an agent is the implementation of its "Perception-Decision-Action" life cycle (cf. fig.6.2), which means a behavior is the execution of a* ***local***[1] *action, consequently to an* ***autonomous*** *decision process, as response to a* ***limited***[2] *perception, which can be an external stimuli/information (reactive behavior) or originated from the agent itself (proactive behavior like spontaneous communication to help the neighbor in the worst situation).*

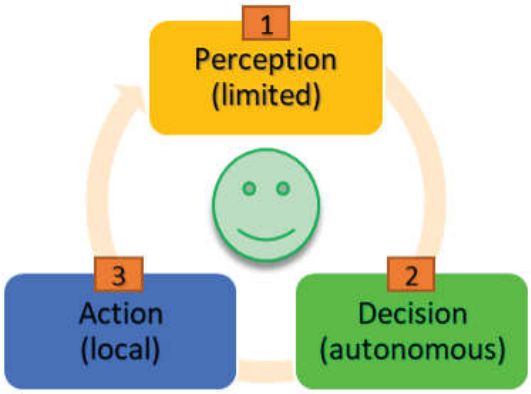

Figure 6.2 — Life cycle of an AMAS agent.

[1] An agent interacts with a small set of other agents it knows, called *"neighbors"*.
[2] An agent can perceive only a small amount of its environment information.

6.1.2.1 Adapt the System by its Parts

In this approach, each part P_i of a system S is considered to achieve a partial function f_{P_i} of the global function f_S (cf. fig.6.3).

f_S is the result of the combination of the partial functions f_{P_i}, noted by the operator "$\circ$". The combination being determined by the current organisation of the parts, we can deduce $f_S = f_{P_1} \circ f_{P_2} \circ ... \circ f_{P_n}$. As generally $f_{P_1} \circ f_{P_2} \neq f_{P_2} \circ f_{P_1}$, by transforming the organisation, the combination of the partial functions is changed and therefore the global function f_S changes.

So, enabling a MAS to self-organise consists in enabling the agent to change inside the organisation. The global function realized is the result of the organisation between agents in the system. This reorganisation technique can be extended with two other related techniques:

- *self-tuning*: parts can modify the parameters defining their behavior,
- *self-evolution*: parts can appear and disappear when needed.

To ensure that the system will generate emergent behaviors [248][249] and to be able to *"control"* this emergence, it is necessary to provide the agents with a local criterion which enables them to self-organise. This requires both a theoretical and engineering framework.

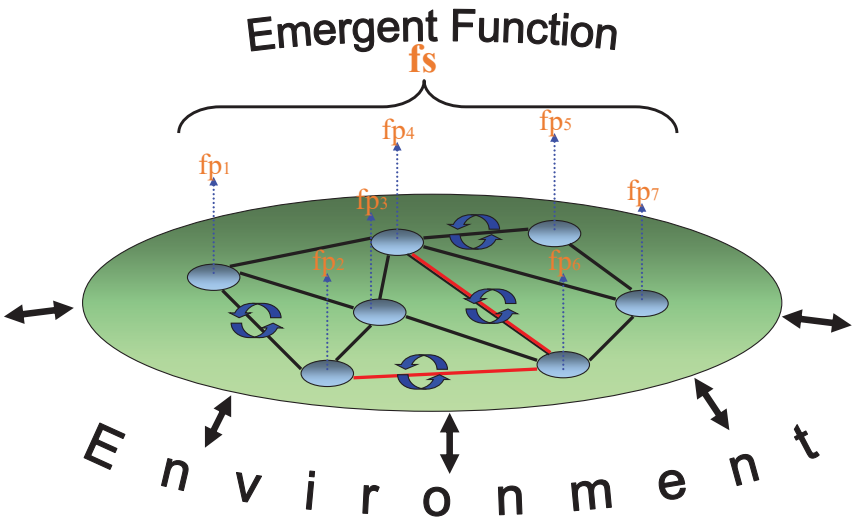

Figure 6.3 — Adaptation: changing the function of the system by changing its organisation.

Definition 4. Cooperation *is the engine of the self-organisation processes taking place in the system and it is the heart of this bottom-up method. Cooperation is classically defined by the fact that two agents work together if they need to share resources or competences [250].*

To this definition is added, the fact that an agent locally tries on one hand, to anticipate problems and on the other hand to detect cooperation failures called *Non Cooperative Situations* (*NCS*, see definition 5) and try to repair these NCS [251].

To anticipate NCSs, the agent always chooses the actions which disturb other agents it knows the less. In others words, the agents, by trying to always have a cooperative attitude, act by reorganising their acquaintances and interactions with the others agents.

Definition 5. *An agent is in a* Non Cooperative Situation (NCS) *when:* $(\neg c_{per})$ *a perceived signal is not understood or is ambiguous;* $(\neg c_{dec})$ *perceived information does not produce any new decision;* $(\neg c_{act})$ *the consequences of its actions are not useful to others.*

The objective is to design systems that do the best they can when they encounter difficulties. The designer has to describe not only what an agent has to do in order to achieve its goal but also which locally detected situations must be avoided and when they are detected how to suppress them.

6.1.2.2 Behavior of an *AMAS* Agent

A cooperative agent in the *AMAS* theory has four characteristics. First, an agent is autonomous. Secondly, an agent is unaware of the global function of the system; this global function emerges from the agent level (micro level) towards the multi-agent level (macro level). Thirdly, an agent can detect NCSs and acts to return in a cooperative state. And finally, a cooperative agent is not altruistic (it does not always seeks to help the other agents), but benevolent (it seeks to achieve its goal while being cooperative).

Agents have to be able to detect when they are in an NCS and how they can act to come back in a cooperative situation. Agents also always try to stay in a cooperative situation and so the whole system converges to a cooperative state within and with its environment.

The main information an AMAS agent uses for its decision process is a specific measure called *criticality*. This measure represents the state of dissatisfaction or urgency of the agent regarding its local goal.

Each agent is in charge of estimating its own criticality and providing it to its neighboht[3]. The role of this measure is to aggregate into a single comparable value all the relevant indicators regarding the state of the agent.

Having a single indicator of the state of the agent simplifies the reasoning of the agents. In addition, this mechanism has the interesting property of limiting the information transmitted to the others agents, which can be of interest in case of a large distributed systems where data privacy, data volume and computational complexity are issues.

With this additional information, each agent can and has to choose to cooperate with the most critical agent he is aware of. This leads to a very powerful heuristic to cut through a search space so as to drive the system to the expected state, effectively achieving a decentralized process that can be qualified as *emergent collective problem solving*.

This describes the typical decision process of a generic *AMAS* agent. But the NCSs and the actions which could be applied to solve them are not generic: designers have to write

[3] In open and untrusted environments, there exists several mechanisms to tackle uncertainty on exchanged information. This is often the case in System of Systems approaches. Inside a given system where each agent has been designed for the same stakeholder, each agent is assumed to provide the most trustful and accurate information.

their own specific NCS set and related actions for each kind of agent they wish the system to contain. Moreover, designers have the task to provide the agents with adequate means to calculate their criticality. But the main idea here is that this is far more manageable and realistic at the local level of each agent (micro level) than at the global level of the whole complex system (macro level).

6.2 Architecture of AMAS4BigData

The framework AMAS4BigData, as introduced in the previous chapter, is a dynamic big data analytics framework based on the AMAS theory. Thence, to describe the architecture of the framework, I present the agents taxonomy with their attributes, their nominal and cooperative behaviors, and the criticality function. However before that, I give the motivations and insights that lead me to design the architecture of AMAS4BigData.

6.2.1 Motivations & Insights

First of all as discussed in the previous chapter (cf. section 5.2), the digital universe is a complex system, wherein an entity is a data source (an IoT sensor, a database attribute, a dataset feature…). These entities can appear and disappear anytime (open system), thence the framework AMAS4BigData must incorporate the corresponding data source to the analytics process, and remove them from it, automatically on the fly without rebooting the systems to update it (continuous adaptation).

The openness feature of the digital universe rises another constraint, the analytics system must have the potential to infinitely expand. However, the scaling of the processing and storage hardware are either hard or costly (see the comparison of scaling techniques 3.2.2.5). So, in order to avoid processing all the data in the same "must to scale" place, like a cloud platform, AMAS4BigData distributes the analytics process over the data source. In other words, it pushes the processing of the data directly to their sources. This edge disturbed processing provides the following advantages:

- Allowing the analytics system to grow automatically with the growth the digital universe (data sources), leading to an elastic[4] computing model.
- Granting to the data provider the power see and control how its data are processed, hence preserving its privacy and insuring its data safety.
- Handling heterogeneous data sources, through expressing the dedicated sub-analytics process for each data at its source, if needed.

In addition, with the recent technological improvements, the data generative devices and applications (the data sources) tend to produce data as continuous streams, whose content evolves over time (dynamic system). Thus, AMAS4BigData has to process these data

[4] Ability to grow and shrink following the available computing power.

streams as soon as they provide a new data (real-time system). Therefore, the framework doesn't require to store the data, avoiding the related storage issues.

Furthermore, when a data stream is unable to be processed, AMAS4BigData has to either load its processing task somewhere else (dynamic load-balancing), consequently to a lack of processing power, or isolate the data stream from the system, if its source is corrupted or faulty. As a result, the framework AMAS4BigData is computing-fault tolerant.

To sum up, the framework AMAS4BigData enable the design of analytics systems that fulfill the requirements of dynamic big data analytics (cf. section 5.1), thanks to the features they gain inherit from the framework:

- non-centralized, elastic[4] and dynamic analytics process;
- continuous adaptation to the digital universe and real-time processing of heterogeneous data;
- No storage, privacy and safety insurance, of the data.

6.2.2 Architecture Overview

AMAS4BigData is a generic framework for online[5] processing of data streams. Its architecture is designed as a dynamic analytics network (graph) (cf. 6.4), rather than a cyclic pipeline, **wherein nodes as well as edges can appear and disappear at anytime**. Hereafter, I give the description of the nodes and links.

6.2.2.1 Nodes

Nodes (vertices) represent the data inputs (data sources) in a one-to-one correspondence (or bijection) i.e. one node represents one input.

Analytically speaking, first they *pre-process* the raw data streams generated from the inputs. Then, send them to their neighbors through their analytic links (edges). Finally, they *exploit* the information they get from these analytic links to build and update on the fly an individual (partial) knowledge, which they gather for a collective (global) knowledge.

6.2.2.2 Links

Links (edges) are the embodiment of the analytic bounds between two nodes. They continuously transform the data originating from one node to information for the second one and can be seen as partial data analysis pipes. For this reason, I designate them as *"analytic links"*. These links are created and removed as needed by the analytics process, and have different states, as described bellow.

An existing analytic link can be either *active* or *inactive*: it is active if it has a computing resource to carry out its analytic work, it is inactive otherwise. Furthermore, a link stays alive (active or not) as long as it is useful for the analytics process. The link usefulness

[5] Or *at runtime*: Process the data one by one as it is produced, and not as whole (batch).

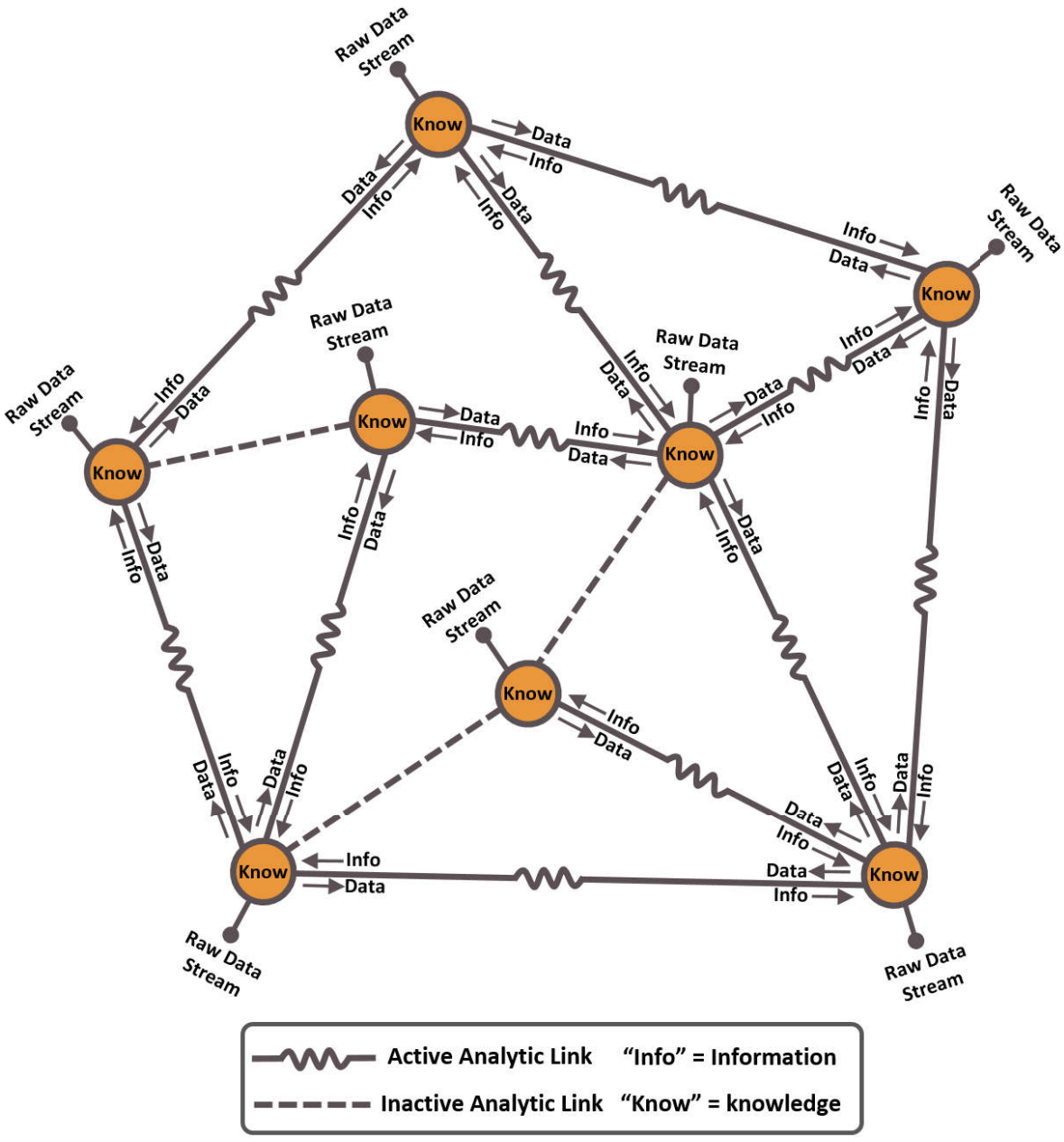

Figure 6.4 — An example of AMAS4BigData architecture.

means it either tries to find relevant information from the data it carries (*weak* link), or it already found information and tries to find new ones (*strong* link).

The exhaustive life cycle of an analytic link is described in the following algorithm (6.1), and illustrated with an automaton in figure 6.5.

First, a link is considered *weak*, searching for useful information. Then after a given time (amount of processed data) *LinksRemovalTime*, if the link usefulness evaluation is not high enough (*min_eval*) it becomes useless and it is removed. Else, the link becomes *strong* and stays alive and active as long as the link evaluation remains high, if not the link becomes *weak* again.

Algorithm 6.1: Links life cycle

Input: the *data* to be processed through the link

```
1.  State ← weak;                                   /* initialize the link state */
2.  RemovalStep ← LinksRemovalTime; /* initialize the RemovalStep of the link
    to LinksRemovalTime (by default ) */
3.  Step = 0;                                       /* initialize the Step counter */
4.  forall the data do
5.      Eval ← LinkEvaluation();                    /* evaluate the link usefulness */
6.      if State = weak then
7.          if Step = RemovalStep      /* the RemovalStep is reached, because the
            link did not produce useful information */
8.          then
9.              Remove the link;
10.         else if Eval ≥ min_eval        /* the link becomes strongly useful */
11.         then
12.             State ← strong;
13.             DeactivationStep ← Step + LinksDeactivationTime;        /* set the
                DeactivationStep so the link will deactivate in
                LinksDeactivationTime from the current Step */
14.     else
15.         if Step = DeactivationStep   /* the DeactivationStep is reached, meaning
            the link continuously produced the same information (local
            optimum) */
16.         then
17.             Deactivate the link;
18.         else if Eval < min_eval          /* the link becomes weakly useful */
19.         then
20.             State ← weak;
21.             RemovalStep ← Step + LinksRemovalTime    /* update the RemovalStep
                so the link will deactivate in LinksRemovalTime from now */
22.     Step ← Step + 1;
```

Moreover, if the link remains *strong* after another given time (*LinksDeactivationTime*) it becomes *redundant*, because it produces the same information without novelty. So, the associated computing resource deactivates the link (but the information produced so far has been sent and saved in the related nodes) in order to leave this local optimum and explore new regions (links) of the data space.

The evaluation of the analytic links is a metric or a function related to the goal of the analytics process, which is expressed by the user of AMAS4BigData while designing an analytics tool.

For example, if the analytics system was designed for mining dynamic association rules,

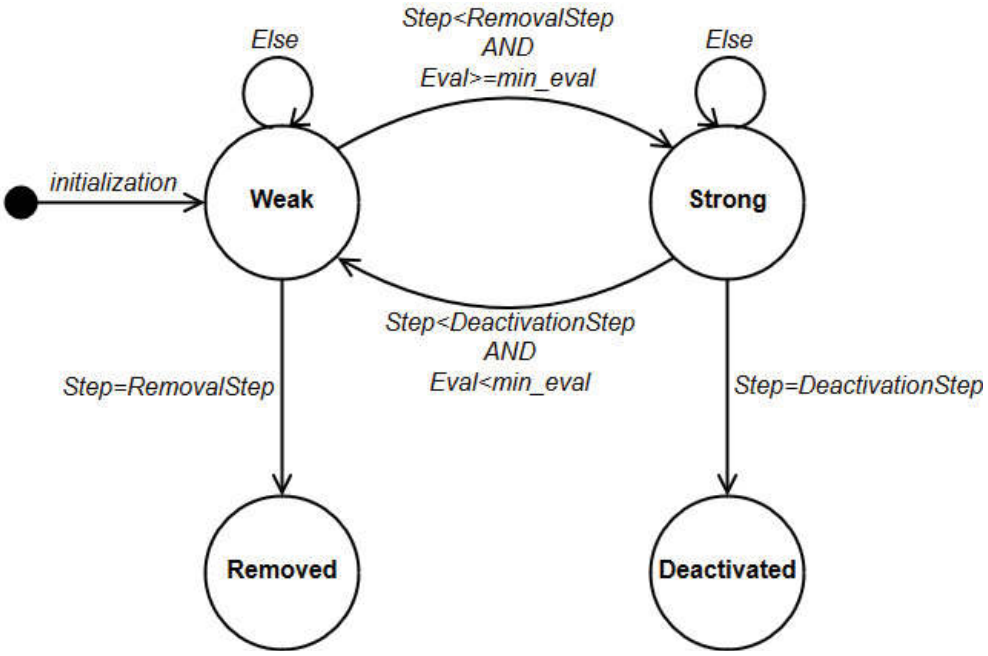

Figure 6.5 — Analytics links life cycle automaton.

a node represents one item and handles all the transactions that contain it, and an analytic link incrementally computes (online) the *confidence* of the two possible association rules composed of the two related nodes (items) A and B. Thence in this case, the evaluation of a link is the *confidence* of the related association rule.

Therefore, a rule $(A \Rightarrow B)$, meaning that each transaction containing the item A also contains B, becomes *strong* when $\#\#AB \geq \#\#A * min\_confidence$ for $LinksRemovalTime$ number of transactions read, where $\#\#AB$ is the frequency of the itemset $\{A, B\}$ and $\#\#A$ is the frequency of the itemset $\{A\}$.

6.2.2.3 Exploration Mechanisms: Exploration by Cooperation

Since cooperation is the heart of the AMAS technology (def. 4), it is by extension the heart of this new framework. Thus as defined, cooperation is achieved by setting local (individual) rules that allow and promote the agents (framework entities) to work together and help each other, when it is best suited.

Therefore, these cooperation rules, in the scope of big data analytics, involve at least two agents that share some of their local knowledge and/or resources of the analytics process, to efficiently explore the data space. More specifically, AMAS4BigData aims at using limited computing power, which can be extended to handle more data. However, for an analytics system with n inputs (data streams), it takes $n(n-1)/2$ analytics links to examine all the possible data combination. This corresponds to a high complexity of $O(n^2)$ that prevents the analytics system to scale up, in a broad sense.

Consequently, the AMAS4BigData framework first assesses the most *critical* nodes (cf. 6.3.4), then allocates them computing resources to create new analytic links. The selected

nodes use one of the following three exploration mechanisms (a set of cooperation rules) to decide which links should be created, in order to analyze a data streams couple only when it is useful for the analytics process.

In addition, because several nodes can apply these three exploration mechanisms simultaneously, they need to coordinate their efforts. This is included in their behaviors described later (6.3.2) to avoid redundancies and overlapping. Also, the concurrent application of the exploration mechanisms may rapidly increase the number of analytic links, thence increasing the system complexity and its need for computing power.

As a consequence, an exploration mechanism is triggered according to a given probability for each of the three mechanisms: P_{TR}, P_{SL} and P_{RL} respectively to *Triangulation*, *Split-Linking* and *Random Linking*. Note that these probabilities are independent and reflect the exploration strategy.

- **Triangulation**: when applied following the probability P_{TR}, aims to form triangles of analytic links by linking two nodes that already have one neighbor in common. For example as illustrated in figure 6.6, given two linked nodes A and B, when a node C link itself to A then A help C to link with B.

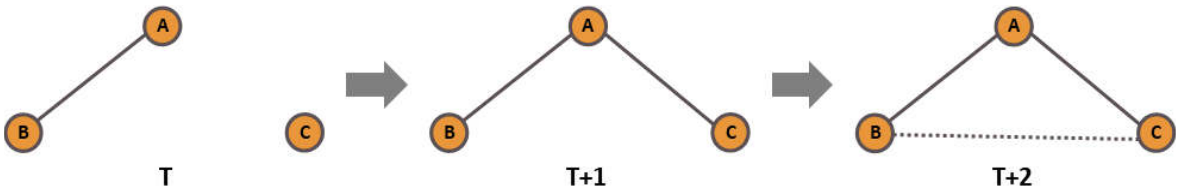

Figure 6.6 — Triangulation mechanism.

  The triangulation mechanism tends to produce clusters of highly connected nodes, because when two nodes are indirectly related by a transition node, the analytic link between the two is most likely useful as well;

- **Split-Linking**: triggered according to the probability P_{SL}. Given a node X and a set L of links related to X and created at the same time, this mechanism is applied when at least one link of L becomes *strong* and the others do not. As a consequence, X destroys the *weak* links and inform them that they should link their related nodes together, as illustrated in fig.6.7, because they might be alike and a link between them useful.

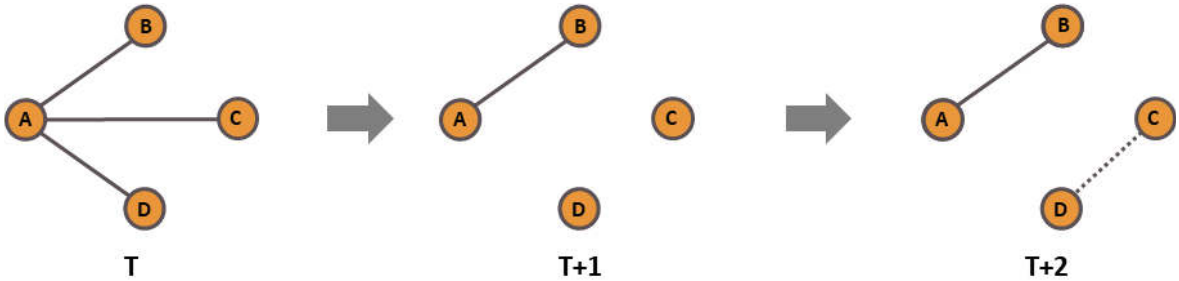

Figure 6.7 — Split-Linking mechanism.

The split-linking mechanism, like the triangulation mechanism, tends to form dense clusters of nodes that might lead to *strong* links which produce useful information to analytics process. This exploration mechanism is also a *relaxation*[6] technique since it shift the task of discovering useful information from the original node to its neighbors and thence exploring new regions (links) of the data space;

- **Random Linking**: is a naive, yet necessary, exploration mechanism that chooses randomly at most $X\%$ (X_factor) of the non-explored nodes to create links with them, in order to leave local optima or initialize the analytics process (see figure 6.8).

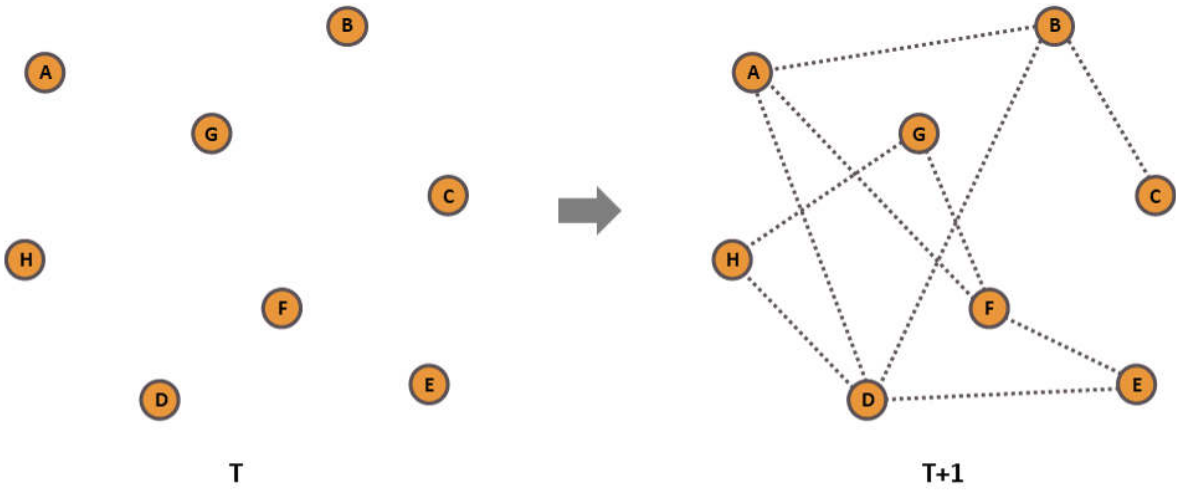

Figure 6.8 — Random Linking mechanism at the initialization phase.

In this case (analytics process), a local optimum is a situation that produces no (new) valuable information, meaning most or all of the analytic links became useless, or that produces the same information repeatedly, hence after a given amount of time the knowledge of a node does not change anymore.

Moreover, when the previous mechanisms can't produce new links, the random linking does it instead as exploring new links is usually important.

The remaining parts of this chapter describe the *Adaptive Multi-Agent Systems (AMAS)* used to design this new dynamic big data analytics framework.

6.2.3 Agents Taxonomy

In order to comply with its theoretical architecture presented previously (cf. 6.2.2), AMAS4BigData is composed of many agents that are of three type: *Percept* agent (as they perceive the data) or *Compute* agent (as they handle the computational tasks of the analytics process), and *Access* agent (essentially a directory and utilitarian role). I give hereafter the general description of the agents, their roles and attributes.

[6] Approximation of a difficult problem by a nearby problem that is easier to solve.

6.2.3.1 Percept Agent

A percept agent, or "percept" for short, is the *agentification*[7] of a node in the AMAS4BigData architecture (cf. 6.2.2.1). As such, a percept is an agent plugged to a unique data source that *perceives* (read or receive) its raw data stream, cleans the data and sends them to its related compute agents, then constructs the global result of the analytics process in conjunction with the other percepts.

A percept is also responsible for managing its neighborhood[8], by autonomously and cooperatively choosing when to link itself, to which other percept to link itself, and when to remove some links. Furthermore, the percept helps other percepts to link with each others when it is relevant for the analytics process, using the exploration mechanisms (cf. 6.2.2.3).

At the agents level (micro level), a link is the embodiment of the interaction between two percepts that collaborate to dynamically construct and display the global data model. The aggregated dynamic links of all the percepts, build a dynamic interaction graph that represents the state of the analytics system.

The percepts agents are characterized with the following attributes:

Attribute	Description
ID	a unique alpha-numeric key to identify the agent. **E.g.** *"23568841-e1db-4218-aac3-d7d300d9997f"*.
Name	the label of the agent and considering that a precept represents a data source, the name of a percept is the designation of its data source. **E.g.** *"thermometer_classroom_301"*.
SpawningTimestamp	the timestamp (date and time) of the moment the agent was spawned. **E.g.** *"1528279912"* which corresponds to *"2018-06-06_10:11:52"*.
Neighbors	the set of the percepts (<*ID,Name*>) that are linked to this percept. **Initially**, *it is randomly filled with some of the other percepts (cf. 6.2.2.3)*. **E.g.** *"{<7537662d-f3d9-446f-8db3-da18571674a1,luminosity_classroom_301>}"*.
PreviousNeighbors	the set of the percepts (<*ID,Name*>) that were previously linked to this percept. **Initially**, *it is empty*.
DataConsumers	the set of the compute agents (<*ID,LinkName*>) that must receive the percept data to process (consume) them; put in another way it represents the active links of the percept. **Initially**, *it is randomly filled with some randomly allocated compute agents*. **E.g.** *"{<0f74cce7-95b2-4bfc-a28f-6df92116f0e4,luminosity_classroom_301 -thermometer_classroom_301>}"*.
Links	the set of all the related links (<*LinkName, LinkTimestamp*>). **Initially**, *it is filled with the information related to the Neighbors*. **E.g.** *"{<luminosity_classroom_301-thermometer_classroom_301,1528279917>}"*.

[7] The representation of one entity with an agent.
[8] The set of agents (neighbors) with which an agent interacts.

InactiveLinks Archive	is a set of the inactive analytic links with their last evaluation (<*LinkName,Eval*>), so it can be sent to a future compute agent for reactivating an inactive link. **Initially**, *it is empty*. **E.g.** *"{<luminodity_classroom_301-thermometer_classroom_301,0.9>}"*.
UsefulLinks	the set of the links (active and inactive) that are considered as strong. **Initially**, *it is empty*.
LinksQueries	the queue (FIFO set) of the links to analyze first. **Initially**, *it is empty*.
Criticality	an evaluation of the situation of the percept, computed with the *criticality computation* function defined later (cf. 6.3.4). **Initially**, *it is set to 1, and updated at the end of the first analytics cycle. So, the percepts with the least amount of computing resources become the most critical*.
ConnectionDegree	the number of percepts related through the analytic links; designated also as *"neighborhood size"*. **Initially**, *it is set with the number of random links established formerly.*
TimeSinceLast Allocation	the amount of time (number of cycles) elapsed since the last compute agent allocation; this is needed to compute the percept criticality. **Initially**, *it is set to 0*.
TimeSinceLast Linking	the amount of time (number of cycles) elapsed since the last creation of a new analytic link; it is also required to compute the criticality. **Initially**, *it is set to 0*.
TimeSinceLast LinkValidation	the amount of time (number of cycles) elapsed since the last validation of a an analytic link (becoming strong); the percept uses it to know when to apply the *RandomLinking* behavior (cf. 6.3.2.2), because it reached a local optimum in the data space. **Initially**, *it is set to 0*.
AllocatedResources	is the number of all the computing resources (compute agents) allocated to the percept over time **Initially**, *it is set with size of the DataConsumers*.

Table 6.1: Percept agents attributes.

6.2.3.2 Compute Agent

A compute agent is related to two percepts and is in charge of analyzing the link between them. More specifically, a compute agent represents an available computing resource, which is provided by any data source (data generative device) that has some computing power or any other available machine.

A compute agent is in charge of carrying out the computational tasks of the analytics system. It chooses the most critical percept (the percept that needs the most a computing resource according to the criticality function defined in 6.3.4) and provides it with the power to activate (analyze) a new link or reactivate (resume the analysis) an inactive link.

The number of these agents changes over time with respect to the available computing power and the analytics process needs:

- On one hand, when a compute agent decides that it is not useful anymore to the analytics system, meaning after a given amount of time there are no more computational tasks to carry out, it destroys itself in order to free the corresponding computing power for another system (scale down).
- On the other hand, compute agents can be spawned (created) by percepts when there is enough computing power available and the number of the overall analytics links to be analyzed is far greater than the current number of compute agents (scale up).

As defined here, a compute agent is an abstract agent and it becomes concrete when a function to be computed is implemented (defined) within it.

The attributes of the compute agents are detailed in the following table:

Attribute	Description
ID	a unique alpha-numeric key to identify the agent. **E.g.** *"271300c3-5716-4ea9-8cdc-f09801971396"*.
SpawningTimestamp	the timestamp (date and time) of the moment the compute agent is spawned. **E.g.** *"1528279930"* which corresponds to *"2018-06-06_10:12:10"*.
Number	the number of the current compute agents, which increases each time a new one is spawned. **E.g.** *it is set to 1 if it is the first comute agent spawned*.
LinkName	the name of the analytic link carried out by the compute agent; it is the concatenation of the alphabetically sorted names of the two related percepts. **E.g.** *"luminosity_classroom_301-thermometer_classroom_301"*.
URI	the Uniform Resource Identifier string of the real computing resource (machine, device...) that supports the agent. **E.g.** *"141.115.73.23"*.
Percepts	the set of the two related percepts (<*ID*,*Name*>). **E.g.** *"{<23568841-e1db-4218-aac3-d7d300d9997f,thermometer_classroom_301>, <7537662d-f3d9-446f-8db3-da18571674a1,luminosity_classroom_301>}"*.
Free	a boolean that shows if the compute agent is free or allocated to analyze the link between the *Percepts*. **Initially**, *it is true*.
Eval	the evaluation of the associated analytic link. **Initially**, *it is set to 0*.
State	represents the usefulness of the analytic link, which is either *weak* or *strong*. **Initially**, *it is weak*.

TimeSinceLast Allocation	the amount of time (number of cycles) elapsed since the last allocation to a percept. **Initially**, *it is set to 0*.

Table 6.2: Compute agents attributes.

6.2.3.3 Access Agent

The access agent is unique, but it can be re-spawned in case of failure, and it can be replicated to duplicate the information it contains in order to increase the number of concurrent accesses, thence avoiding bottlenecks.

The access agent doesn't take part in the data processing, meaning it has no control over the analytics process. So, there is no centralization. However, it helps the analytics process thanks to two roles:

- **Shared Directory**: is a common log file and directory accessible by all the agents. Each agent can register/unregister itself in the directory or record its activity in the log file. For instance, It is useful for the percepts to link themselves to unknown percepts for them, but registered in the *access agent*.

  It also serves as a Blackboard [252] for indirect agents communication of their information (attributes) required for the computation of their criticality function (cf. 6.3.4).

- **Interactive Node**: allows the end-user[9] to interact with the system and execute some basic queries:

  - Getting the log, the list/number of the percepts (data sources), links (active and/or inactive), resources, processed data;
  - Adding/removing a data source and, by extension, spawning/destroying a percept agent that represents it;
  - Adding/remove a computing resource;
  - Increasing/decreasing the criticality of a percept;
  - Overriding the system parameters (cf. 6.4);
  - Feedback integration by selecting links to activate/reactivate and/or editing a black list (links not to be analyzed). In either case, the access agent conveys the list of these links to their related percepts so that they can analyze them or not, corresponding to the user feedback.

From these functionalities, I derived the attributes of the access agent:

[9] User of the AMAS4BigData based analytics tool, whether it might be a human or a software.

Attribute	Description
ID	a unique alpha-numeric key to identify the agent. **E.g.** *"42c6db83-d564-4ba8-8b5e-ad978440d24f"*.
PerceptAgentsDirectory	the list of the percepts with their information (<*ID*, *Name*, *SpawningTimestamp*, *Criticality*, *#UsefulLinks*, *#ActiveLinks*, *ConnectionDegree*, *TimeSinceLastAllocation*, *TimeSinceLastLinking*). **Initially**, *it is filled with the information of the percepts agents created from the first input data streams*.
ComputeAgentsDirectory	the list of all the compute agents (free and allocated) with their information (<*ID, Name, SpawningTimestamp, URI*>). **Initially**, *it is filled with the information of the first compute agents*.
ActiveLinksDirectory	the list of all the active links in the system. **Initially**, *it is filled with the first links randomly activated compute agents*.
InactiveLinksDirectory	the list of all the inactive links in the system. **Initially**, *it is filled with the first inactive links*.
ComputingResources	the list of all the computing resources provided for the system and their load[10] (<*URI, load*>). **E.g.** *"{<141.115.73.23,0.1>}"*.

Table 6.3: Access Agent attributes.

After the description of the static aspect of the AMAS4BigData framework, let us now describe the dynamic aspect of the framework.

6.3 Framework Functioning

This section focuses on how AMAS4BigData works (function) through the description of the agents nominal behaviors (when they work normally without issues) and cooperative behaviors (when the agents encounter NCS and try to fix them).

But first, I give a rundown of the framework functioning for the sake of a better understanding of the detailed description after that.

6.3.1 Functioning Overview

As explained earlier, the AMAS4BigData framework takes as input several data streams, like the data produced by a network of sensors (IoT). The framework functioning is divided into an *initialization phase* that describes how the framework is set, and a *nominal functioning* portraying how it works afterwards.

[10]The ratio between the number of compute agents and the overall computing power of the resource.

6.3.1.1 Initialization Phase

It starts by spawning (creating) an access agent that virtually encompasses the whole analysis and serves as an indirect interaction medium (blackboard) for the agents. The direct interaction is ensured via message sending. Then the framework initiates the analytics process by means of the input streams agentification. In other words, one percept agent is created and registered for each data stream in order to represented them and handle their processing inside the framework.

In addition, an initial pool of compute agents $\#ComputeAgents$ is created based on the X_factor and the number of input data streams n, as follows:

$$\#ComputeAgents = X\_factor * \frac{n(n-1)}{2} \tag{6.1}$$

The second to last step, of the framework initialization, consists in building randomly a first neighborhood for each percept, meaning the percepts set up analytic links between each others randomly thanks to the *random linking* mechanism (cf. 6.2.2.3). When it is done, the initialization is completed by allocating the compute agents to analytic links at random.

6.3.1.2 Nominal Functioning

From the data processing point of view, every time a data streams produces a new data, its corresponding percept agent perceives it (reads it or receives it). The percept pre-processes the data and sends it to its neighbors via their relating links.

When a link receives the data from its two related percepts, its compute agent transforms the data and attempts to find, then sends the discovered useful information to each percept. Moreover, the analytics links updates their status following their life cycle (cf. 6.2.2.2). Once the information are obtained, the percept agents update their local knowledge and work together to collectively build the global knowledge.

As for the agents organisation, the percepts exploit their knowledge to better explore the data space by modifying their neighborhoods (changing their links), using the exploration mechanisms defined previously (cf. 6.2.2.3). Also, the free compute agents are assigned to the most critical links, as explained later (cf. 6.3.4).

In the next parts, I detail the framework functioning (dynamic aspect) with a thorough description of the agents life cycles and their corresponding behaviors. Furthermore, the interaction of the agents behaviors is explained and illustrated at the end of this chapter (cf. 6.3.5), in order to get an extensive overview of the framework functioning.

6.3.2 Agents Life Cycles & Behaviors

All the agents, except for the *access* agent, have some common behaviors (*register* and *unregister*). Most of the behaviors are dedicated to one agent type as described here, following the "Perception Decision Action" agents life cycle (def.3). Furthermore, an agent behavior can be either:

- *reactive*: like a biological reflex, the agent perceives an external stimulus (information, message...) from its environment and executes an action by respect of a set of defined rules, whose conditions are based on the stimuli. For instance, when birds fly as a swarm/flock they follow reactive rules of proximity and alignment.
- *proactive*: the agent triggers a behavior on its own. It processes its knowledge about itself, and its environment, then decides which action to execute in order to achieve its goal. For example, every day a "stock management" agent checks by itself if there are items out of stock, or estimates how fast they will be sold, then replenishes them if it thinks it is necessary.

The behaviors of the agents, detailed in the tables below, are extracted from their agents life cycles.

6.3.2.1 Percept Agent Life Cycle

Algorithm 6.2: Percept Agent life cycle

1. Initially When a percept is spawned:
2. It registers itself in the access agent, so it can be known by the other agents;
3. It builds a first neighborhood with the *Random Linking* exploration mechanism;
4. Data processing:
5. When it perceives a new raw data, the percept pre-processes it and sends it through its active links (compute agents);
6. Once it receives new information, the percept uses it to build/update the knowledge locally and globally;
7. Links and compute agents management:
8. If the percept is critical (cf. 6.3.4), it gets a free compute agent then sends to it the information of the link with the highest priority amongst the links in stand-by to be activated;
9. When one of its inactive links gets a compute agent, the percept activates the link. In return, when a compute agent leaves a link, its related percepts deactivate it;
10. Once a link becomes useless, it is destroyed and its percepts remove it;
11. At the end of an analytics cycle (step), if a percept needs more computing resources to carry out its inactive links, it spawns (creates) a new compute agent if there is enough computing power available;

12. Links Creation:
13. A percept uses one of the three exploration mechanisms, following their trigger probabilities, to choose a new neighbor;
14. Once the choice of a new neighbor is made, the percept creates the link between them and informs the neighbor of it;
15. When the percept becomes useless (no more data), it unregisters itself and self-destroys;

6.3.2.2 Percept Agent Behaviors

	Behavior	Perception	Decision	Action
Proactive	*Register*	agent spawned	self register in the access agent *PerceptAgentsDirectory* and build an initial neighborhood	send a *percept registration* message to the access agent and execute the *RandomLinking* behavior
	Unregister	agent destroyed	self unregister from the access agent *PerceptAgentsDirectory*	send a *percept unregistration* message to the access agent
	Random Linking cf. 6.2.2.3	Initialization or *TimeSince LastLink Validation = Link Validation Time*[11]	if the probability allows it, access the *PerceptAgents Directory*, select new percepts randomly and create links with them	if $Random(P_{RL})$[12] is true and the *PerceptAgentsDirectory* of the access agent is not empty, choose randomly from it at most X% new percepts that are not in *PreviousNeighbors* and for each one execute the *Linking* behavior
	Triangulation cf. 6.2.2.3	useful link	store the useful link and use it to make triangulations if the probablity allows it	insert the link in *UsefulLinks* and reset *TimeSinceLastLinkValidation*, then if $Random(P_{TR})$ is true, send the related percept to the other useful neighbors
	Split Linking (cf. 6.2.2.3)	useful link	if the probability allows it, destroy the non-useful links that started at the same time as the useful one and link their related percepts	if $Random(P_{SL})$ is true, compute the split-linkng list (useless percepts that have a low *Connec--tionDegree)* and send it to the compute agents associated to the useless links, then execute the *DestroyLink* behavior for each one

[11] *LinkValidationTime* is the time that a percept agent waits before randomly linking with other percepts. By default it is 30 $cycles(steps)$ and can be overridden by the user.

[12] $Random(p)$ is a function that returns a boolean set at true given the probability p.

	Spawn Compute Agent	end of a cycle	if the number of the agent links to be analyzed is high, spawn a compute agent	if $\#LinksQueries > \#Connection\ Degree$, access the shared directories and check if $\#Compute\ AgentsDirectory < \#ActiveLinks\ Directory$, then select the computing resource with the least *Load*. If $Load < 1$, spawn a compute agent.
	Update Attributes	end of a cycle	update *Criticality*, *TimeSince LastAllocation* and *TimeSince LastLinking*	increment *TimeSinceLastAllocation* and *TimeSinceLastLinking*, compute the criticality function and store the result in *Criticality*, then send it to the access agent
	Self Destroy	useless percept	self-destroy	execute *unregister* behavior and self-destroy
Reactive	*Pre-Process Data*	new raw data	send the pre-processed data to all related computing agents	pre-process the data and send it to the *DataConsumers*
	Process Information	new info	process the information to build knowledge	execute the partial analysis method on the information
	Associate Compute Agent	free compute agent	associate the compute agent to either the link that must be analyzed first, if any, or to a new random link	if *LinksQueries* is empty, execute the *RandomLinking* behavior; then, get the first link in *Links Queries*, send it to the compute agent and increment *AllocatedResources*
	Activate Link	allocated compute agent	activate the corresponding link and start sending data to the allocated compute agent	first reset *TimeSinceLastAllocation*, then if *InactiveLinksArchive* contains the link, remove it and send its evaluation to the compute agent, else insert the related link with the current *Timestamp* in*Links*; then, add the compute agent to the *DataConsumers* and the related percept in the *Neighbors*
	Deactivate Link	leaving compute agent	deactivate the corresponding link and stop sending data to the associated compute agent	remove the compute agent from the *DataConsumers* and insert the link and its evaluation in *InactiveLinksArchive*
	Linking	percept	create a link with the percept if not already done	if the percept is not in *Neighbors* insert a new link with the percept in *LinksQueries* and reset *TimeSinceLastLinking*

	Destroy Link	useless link	remove the link and save the related percept	delete the useless link from all link sets, then insert the useless percept in *PreviousNeighbors*, remove it from *Neighbors* and remove the compute agent from the *DataConsumers*

Table 6.4: Percept agents nominal behaviors.

6.3.2.3 Compute Agent Life Cycle

Algorithm 6.3: Compute Agent life cycle

1. First, when a compute agent is spawned, it registers itself in the access agent so it can be known by the other agents;
2. Then **for each** analytics cycle (step),
3. **if** *the agent is* Free **then**
4. it searches for the most critical percept in order to give him the power to activate an analytic link;
5. when the percept chooses a link to activate, the compute agent allocates itself to the link so it can carry out the computational work of the link;
6. **else**
7. when it receives data from its related percepts, it attempts to process it into information and sends it back to the percepts;
8. it also helps the percepts linking to each other when the *split-linking* mechanism (cf. 6.2.2.3) is applied;
9. then, it applies the procedure of the link life cycle (cf. algorithm 6.1), and if the link becomes strong, or weak, or useless, the agent validates it, or leaves it, or removes it respectively;
10. If at least one of its related percept is destroyed, the compute agent destroys the link and set itself free in order to be used by other percepts;
11. When the agent becomes useless, because the corresponding computing resources is withdrawn from the system, or the agent is not used anymore (has been for a long time), it self-unregisters and self-destroys;

6.3.2.4 Compute Agent Behaviors

	Behavior	Perception	Decision	Action
	Register	agent spawned	self register in the access agent *ComputeAgents Directory*	send a *computing agent registration* message to the access agent

Proactive	*Unregister*	agent destroyed	self unregister from the access agent *Compute AgentsDirectory*	send a *computing agent unregistration* message to the access agent
	Search Percept	start of a cycle	If free is true, search for the most critical percept and propose to it the computing resource	if *Free* is true, then look in the access agent *PerceptAgentsDirectory* the most critical one and send it a *free compute agent* message
	Evaluate	data processed	evaluate the analytic link	execute one iteration of the algorithm 6.1 loop then according to the result, execute either *ValidateLink*, *LeaveLink*, or *RemoveLink* behavior
	Validate Link	useful link	notify the percepts	send a *useful link* message to the percepts agents
	Leave Link	weak link	leave the link to analyze another link	send a *leaving compute agent* message with the link (<*LinkName*,*Eval*>) to the percepts and the access agent; then, set *Free* to true
	Remove Link	useless link	remove the link between the percepts and from the access agent, then free the computing resource	send a *useless link* message to the percepts and the access agent, then set *Free* to true
	Release Resource	useless agent	after a long time without allocation, self-destroy	If *Free is true AND TimeSinceLast Allocation* $>$ *MaxIdleDelay*[13], then execute *unregister* behavior and self-destroy
Reactive	*Self Allocate*	link	self-allocate to the link	send an *allocated compute agent* message with (<*ID*,*LinkName*>) to the percepts and the access agent and set *Free* to false
	Process Data	new data	transform the data into information and send it to the percepts	apply the partial analysis method on the data of one percept and send the result information to the other one
	Split Linking (cf. 6.2.2.3)	split linking list	send the split-linking list to the percepts of the same list so they can link to each others then leave the link	send to each percept in the split--linking list the list itself, then execute the *DestroyLink* behavior
	Destroy Link	percept destroyed	destroy the link	set *State* to *strongly_useless* and execute *RemoveLink* behavior

[13]The amount of time, number of cycles to be defined by the user, before self-destruction. By default $MaxIdleDelay = 10 + RandomVal(10)$.

	Self Destroy	withdrawn computing resource	deactivate the link and self-destroy	execute *LeaveLink* behavior and *unregister* behavior, then self-destroy

Table 6.5: Compute agents nominal behaviors.

6.3.2.5 Access Agent Behaviors

The access agent has only reactive behaviors, since it serves as a shared directory and an interactive node, and it doesn't actively contribute to the analytics process.

For this reason, considering the *access* as an agent may be arguable, but it has the advantage of exploiting the mobility and autonomy features of an agent in order to handle the other agents signals (messages and queries), where and when it is possible according to the available computing power.

Moreover, as described (cf. 6.2.3.3) the access agent can self-replicate for workload balancing and be extended with new skills and behaviors if the framework user wants to.

Behavior	Perception	Decision	Action
Register Percept Agent	percept agent registration	register the percept agent	add the percept into the *Percept AgentsDirectory*
Unregister Percept Agent	percept agent unregistration	unregister the percept agent	remove the percept from the *PerceptAgentsDirectory*
Register Compute Agent	compute agent registration	register the compute agent	add the compute agent into the *ComputeAgentsDirectory*
Unregister Compute Agent	compute agent unregistration	unregister the compute agent	remove the compute agent from the *ComputeAgentsDirectory*
Spawn Percept Agent	new data source	create a dedicated percept agent	spawn a new percept agent dedicated to the new data source
Destroy Percept Agent	data source removed	destroy the dedicated percept agent	send a *useless percept* message to the dedicated percept
Spawn Compute Agents	new computing resource	register the computing resource and create new compute agents	add the new computing resource (*URI*) in *ComputingResources* list and spawn new compute agents in proportion to the power of the computing resource

Respawn Compute Agent	compute agent unregistred	randomly spawn a new compute agent if needed and available computing resource	if $\#notLoadedComputingResource > 0$ *and* $\#FreeComputeAgents < 5\%$ *and* $\#ActiveLinks < InactiveLinks$ *and* $Random() = True$ then spawn a compute agent
Destroy Computing Agents	computing resource removed	destroy the related compute agents	send a *withdrawn computing resource* message to each compute agent in the *ResourcesDirectory* that has the same *URI* of the removed computing resource and remove it from the *ComputingResources* list
Activate Link	allocated compute agent	register the associated active link	add the *LinkName* in the *ActiveLinks Directory*, and remove it from the *InactiveLinksDirectory* if it is inside
Deactivate Link	leaving compute agent	unregister the associated active link	add the *LinkName* in the *Inactive LinksDirectory*, and remove it from the *ActiveLinksDirectory*
Remove Link	useless link	remove the link from the directories	remove the link from the *ActiveLinks Directory* and *InactiveLinksDirectory*
Update Criticality	percept criticality	update the percept criticality	replace the previous criticality of the percept inside the *PerceptAgents Directory* with the new one

Table 6.6: Access agent behaviors

It should be noted that the combination of the *ReleaseResource* behavior of the compute agents and the *RespawnComputeAgent* behavior of the access agent, allows the AMAS4BigData framework to self-manage the use of the available computing power, since a compute agent is destroyed and respawned following the data exploration needs.

6.3.3 Non Cooperative Situations & Agents Cooperative Behaviors

Since the agents work concurrently, their interactions (the result of their behaviors) may lead to Non Cooperative Situations (cf. 6.1.2.2). Therefore, I provide AMAS4BigData agents with cooperative behaviors in order to avoid or fix NCSs as described hereafter.

6.3.3.1 Concurrent Linking

- ▷ **Description**: two analytic links are created between the two same percepts during the same cycle (cf. fig.6.9).
- ▷ **Type**: concurrency NCS ($\neg c_{act}$). It means that the result of one agent action is the same as the result of another agent action.

▷ **Cause**: the concurrent execution of the agents exploration mechanisms (cf. fig.6.9)

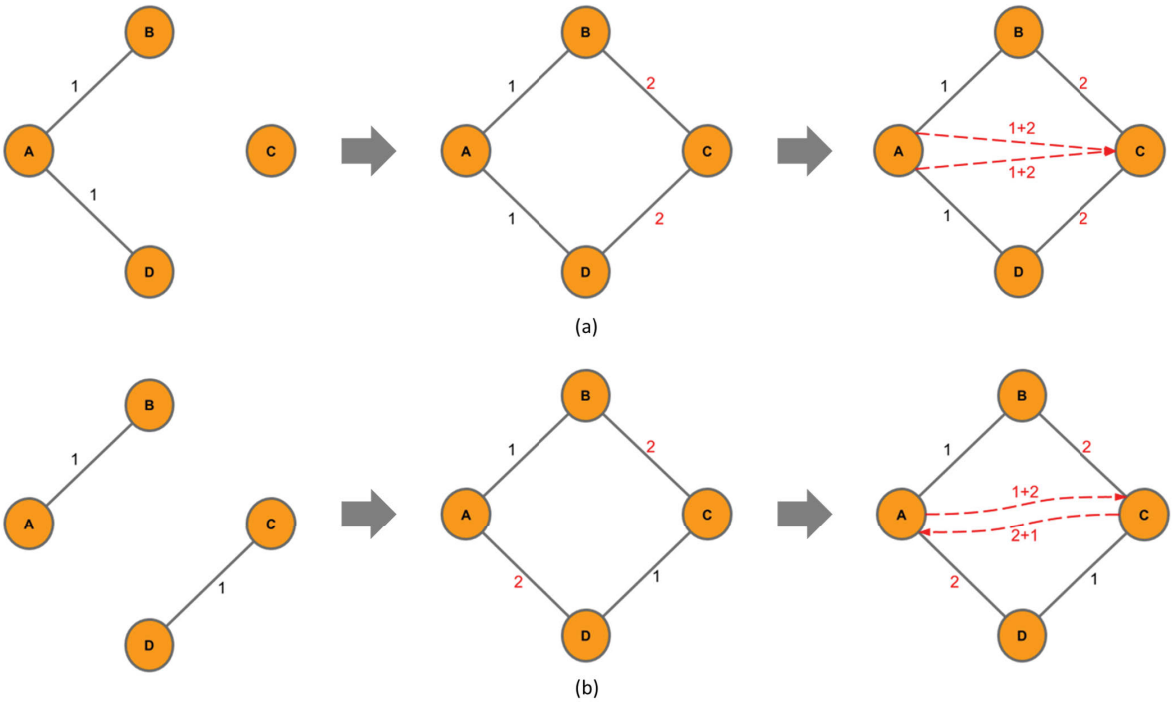

Figure 6.9 — Duplicate Linking NCS.

▷ **Cooperative Behavior**: each percept agent replaces or ignores the duplicate link and the related compute agent is freed. This leads to new cooperative behaviors (cf. table.6.7).

Agent	Behavior	Perception	Decision	Action

Percept	*Remove Duplicate Link*	duplicate link	keep one link and remove the other	*CurrentLink* ← *Links.get*(*NewLink.Name*) ;
				If(*CurrentLink.Timestamp* < *NewLink.Time stamp*) {
				Ignore the *NewLink* ;
				}
				If(*CurrentLink.Timestamp* > *NewLink.Time stamp*) {
				PreviousLink ← *CurrentLink* ;
				CurrentLink ← *NewLink* ;
				Send a *concurrent link* message to the compute agent associated to the *PreviousLink* ;
				}
				If(*CurrentLink.Timestamp* = *NewLink.Time stamp*){
				Ignore both links ;
				Send a *choose duplicate* message to the access agent;
				}

Table 6.7 continued from previous page

Compute	*Leave Concurrent Link*	concurrent link	leave the link	set *Free* to true and execute the *SearchPercept* behavior
Access	*Select Duplicate Link*	choose duplicate	randomly remove one of the links	randomly select one of the links and consider it as duplicate; send a *concurrent link* message to it associated compute agent and send an *allocated compute agent* message to the percepts with the compute agent information related to the non-duplicate link

Table 6.7: Duplicate Linking Cooperative Behaviors.

6.3.3.2 Inactive Percept Agent: Node failure

- **Description**: after a given amount of time (cycles), one compute agent is unable to carry out the analytic work of the associated link, because it lacks data from one percept.
- **Type**: unproductivity NCS ($\neg c_{dec}$). In this case, one compute agent can't produce any result (information) with its perceptions.
- **Cause**: the compute agent receives data from only one percept, due to a failure of the other percept agent, a network failure, or maybe the data source left the analytics system without notification.
- **Cooperative Behavior**: destroy the link and notify the access agent in order to inquire whether the percept doesn't send data anymore, in which case remove it.

  The cooperative behaviors given hereafter are meant to deal with this NCS.

Agent	Behavior	Perception	Decision	Action
Compute	*Destroy Unproductive Link*	one data stream only	destroy the link and notify the access agent	send a *useless link* message to the working percept and send a *not working percept* message to the access agent
Access	*Inquire Inactive Percept*	not working percept	remove the percept if it produces no more data	if the number of *not working percept* messages bound to the percept is equal to the number of its related active links, then send it a *useless percept* message

Table 6.8: Inactive Percept Agent Cooperative Behaviors.

6.3.3.3 Inactive Compute Agent: Link failure

- ▷ **Description**: a percept doesn't receive any information from one compute agent associated to a related link, after a given amount of time (cycle).
- ▷ **Type**: partial unproductivity NCS ($\neg c_{dec}$). In this case, a percept agent produces only a partial result (individual knowledge) with its perceptions.
- ▷ **Cause**: a percept doesn't receive information from one of its related analytic links (compute agent), because of a compute agent failure, a network failure, or by consequence of the *Inactive Percept* NCS (cf. 6.3.3.2) in which case the compute agent can't produce any information.
- ▷ **Cooperative Behavior**: allocate a new compute agent with a different *URI* (from another computing machine) to the dysfunctional analytic link.

6.3.4 Criticality

The criticality translates the need of the percept for a free resource (compute agent) to carry out the analytic work of one related link.

This criticality is computed with a function formulated as a combination of several indicators, which can be considered as metric variating from 0 to 1 where 0 corresponds to non-critical percept and 1 to critical percept.

The criticality indicators are defined hereafter from the most to the least important.

1. **UsefulLinks to ActiveLinks Relative Ration (I_1)**: is the most important indicator, which represents how well a percept p makes use of the resources granted to it (allocated to its links).

   In other words, it measures how productive is the resource allocation for one percept p relatively to all the other percepts c and it is computed as:

   $$I_1^P = \left(\frac{UAR(p)}{\max\limits_{c \in Percepts} UAR(c)} \right)^2 \tag{6.2}$$

   $$UAR(p) = Min(\frac{\#UsefulLinks(p)}{AllocatedResources(p)}, 1) \tag{6.3}$$

   Put it simply, the more the resource allocation strategy of a percept is efficient the more it is critical and gets more resources to allow it to expand its efficient linking.

   However, this indicator alone might lead to a starvation problem, meaning the same percepts always get the available resources and others don't get resources anymore.

   This leads to the next indicator in order to prevent such starvation.

2. **Relative Starvation (I_2)**: as designated this indicator reflects how much a percept p is starving (need of new resources) compared to the other percepts c, as formulated

hereafter:

$$I_2^P = LOG\left(\frac{\#ActiveLinks(p)}{\max\limits_{c \in Percepts} AllocatedResources(c)}\right) \tag{6.4}$$

$$LOG(x) = \frac{log\,(1 + x * 10)}{log(11)} \tag{6.5}$$

3. **Relative Time Since Last Allocation (I_3)**: When several percepts have the same relative starvation, this indicator can help to decide between them, by favouring the percept that did not get any resource for the longest time (number of cycles), formulated as follows:

$$I_3^P = \frac{TimeSinceLastAllocation(p)}{\max\limits_{c \in Percepts} TimeSinceLastAllocation(c)} \tag{6.6}$$

These three first criticality indicators are the main ones involved in computing the criticality of one percept. The following indicators have a minor role and they are designed to refine the criticality computation.

4. **Relative Time Since Last Linking (I_4)**: this indicator promotes the percepts that linked themselves to others the earliest. It is given as:

$$I_4^P = \frac{TimeSinceLastLinking(p)}{\max\limits_{c \in Percepts} TimeSinceLastLinking(c)} \tag{6.7}$$

5. **Relative Connection Degree (I_5)**: expresses the promotion of the percepts with the least number of links (*ConnectionDegree*), according to this formula:

$$I_5^P = sigmoid\left(1 - \frac{ConnectionDegree(p)}{\max\limits_{c \in Percepts} ConnectionDegree(c)}\right) \tag{6.8}$$

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{6.9}$$

6. **Precision (I_6)**: indicates the proportion of the useful links among all the percept links, active or not. It is similar to the concept of precision in *Information Retrieval* field. So it favours the percepts with the best linking strategy and is formulated as this:

$$I_6^P = LOG\left(\frac{\#UsefulLinks(p)}{\#ConnectionDegree(p)}\right) \tag{6.10}$$

7. **Relative Recall (I_7)**: measures how productive a percept p is, relatively to the other percepts c. It highlights the percepts with the least useful links in order to allow them to explore and discover new useful links, following this formulae:

$$I_7^P = LOG\left(1 - \frac{\#UsefulLinks(p)}{\max\limits_{c \in Percepts} \#UsefulLinks(c)}\right) \tag{6.11}$$

Furthermore, to express the importance of each indicators, weights are associated to each indicator resulting to the overall criticality function, as expressed hereafter:

$$Criticality(p) = \sum_{i=1}^{\#I} w_i * I_i^p \tag{6.12}$$

As described previously (cf. 6.4. Framework Use) the AMAS4BigData framework user can override its parameters, thence the user can also override the w_i weights and add new indicators to the criticality function, in order to optimize the implemented analytics system (cf. 7.2.6.2.Criticality Function Extension).

6.3.5 One Analytics Cycle Scenario

Now that the agents behaviors and criticality have been described, let us see how they allow the agents to cooperate during one nominal analytics cycle, as schematized with the sequence diagrams 6.10 to 6.17.

An analytics cycle starts by receiving new data from the sensors, which are perceived by the percepts, pre-processed and sent through the already established analytics links (*action 2*). At the same (*action 1*), the remaining free (not used) compute agents tries to be useful by searching for the most critical percepts (*action RA1*) and associate to them (*action RA2*). Then, these percepts send back a link to be analyzed, or randomly create (*action RA3*) one and send it (*action RA4*). Therefore, the compute agents establish the link between the two linked percepts (*action RA5*).

Once the in-use compute agents get the data from both their associated percepts (*action 3*), they process them and attempts to produce useful information and send them back to the percepts (*action 4*), so they can build and update their knowledge. After that, they evaluate their links (*action 5*). Therefore, they either do nothing, or validate the links (*action 6*) and inform the percepts (*action 7*), or remove the links (*action 10*), or deactivate them (*action 11*) following the links life cycle (cf. 6.2.2.2).

If a link is validated (becomes strongly useful), the percepts tries to create new links thanks to the *triangulation* mechanism (*action 8*) and the *split-linking* mechanism (*action 9*) with the help of their associated compute agents (*action SL2 & SL3*).

At the end of the analytics cycle, the useless compute agents release their computing resource (*action 12*), the useless percepts establish new random links (*action 13*), and each percept that requires more compute agents that available it attempts to spawn a new one (*action 14 & 15*). Finally, all the percepts updates their attributes (cf. 6.4).

To recap, one analytics cycle begins when the framework agents perceive new raw data, then process them according to the agents behaviors, which leads to the system re-organisation by destroying, changing, or creating some links, in order to adapt to the data and produce the most useful and accurate knowledge.

6.4 Framework Use: Implementation of Analytics Systems

To produce an analytics system with the AMAS4BigData framework, a user has to define: the pre-processing method carried out by the nodes (percepts), the data transformation method of the analytic links between the nodes, and the function that evaluates the usefulness of an analytic link.

In addition, the user of the framework can optionally choose (override) new values for the framework parameters:

- the *min_eval* threshold *LinksRemovalTime* and *LinksDeactivationTime* of links life cycle (cf. algorithm 6.1). By default, $min\_eval = 0$, $LinksRemovalTime = 50$, $LinksDeactivationTime = 30$;
- the X_factor of the *RandomLinking* behavior (cf. 6.2.2.3). By default, $X\_factor = 30\%$;
- the trigger probabilities of the exploration mechanisms P_{TR}, P_{SL} and P_{RL} (cf. 6.2.2.3). By default, $P_{TR} = 10^{-1}$, $P_{SL} = 10^{-1}$, $P_{RL} = 10^{-5}$;
- the *LinkValidationTime* of the percept agents (cf. 6.4). By default, $LinkValidationTime = 30$;
- the *MaxIdleDelay* of the compute agents (cf. 6.5). By default, $MaxIdleDelay = 10 + RandomVal(10)$.

Furthermore, the user can guide the current exploration mechanisms (cf.6.2.3.3) and define new ones (cf. 7.2.6.1) for the sake of more efficient analytics process.

The implementation of the AMAS4BigData framework is the subject of the next chapter, wherein a thorough description of one analytics tool based on AMAS4BigData is given and other tools are presented concisely (cf. 7.4).

6.5 Framework Contributions

Thanks to the AMAS theory I was able to design a new big data analytics framework, whose architecture comes in form of a dynamic analytics graph rather than the conventional pipelines, which exhibits interesting features: real-time data processing (no storage required), non-centralization (distributed control and decision which avoid bottlenecks), openness (data streams can be added and removed on the fly), elasticity (adaptive scaling up/down and dynamic work-load balancing), smart exploration and exploitation of the data space, and fault tolerance through the agents cooperative behaviors.

The AMAS4BigData framework allows its users to build new analytics systems that fulfill the dynamic big data analytics requirements, as a result of the framework intrinsic characteristics presented above.

The next chapter gives a thorough example of the implementation of one such dynamic analytics systems.

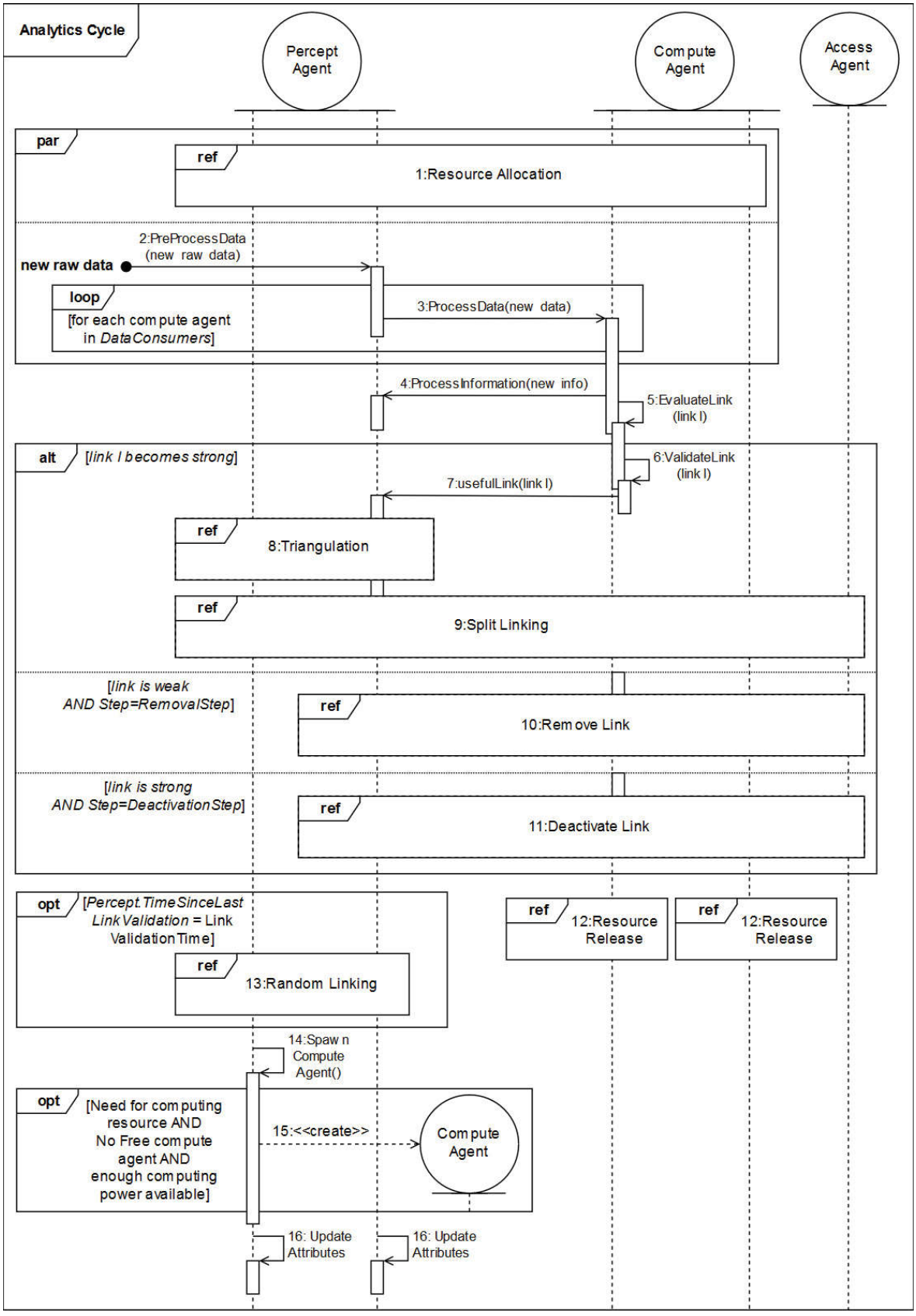

Figure 6.10 — One Analytics Cycle Diagram.

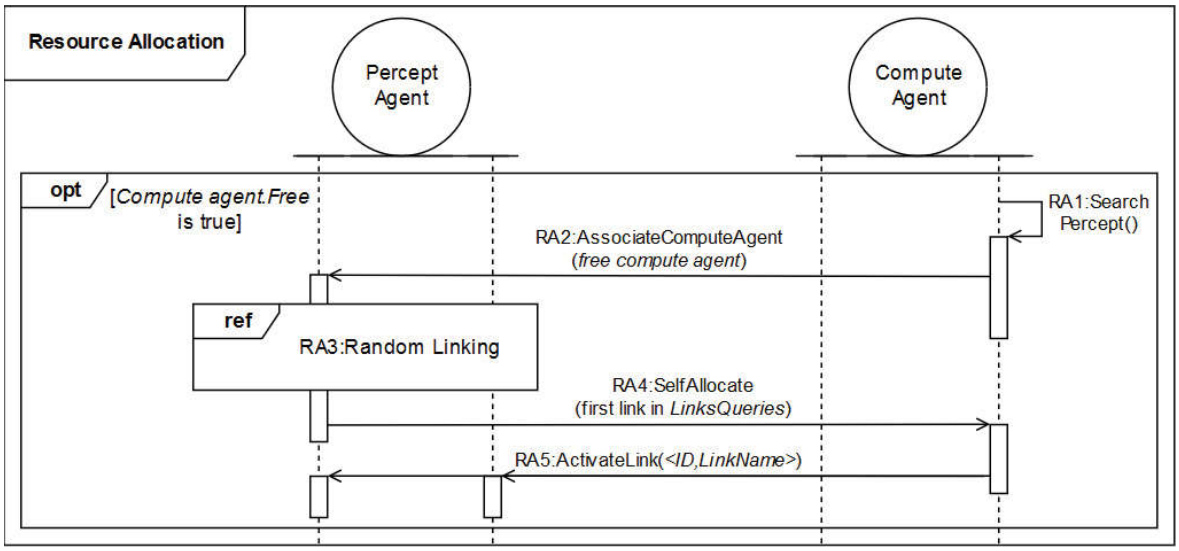

Figure 6.11 — Resource Allocation Diagram.

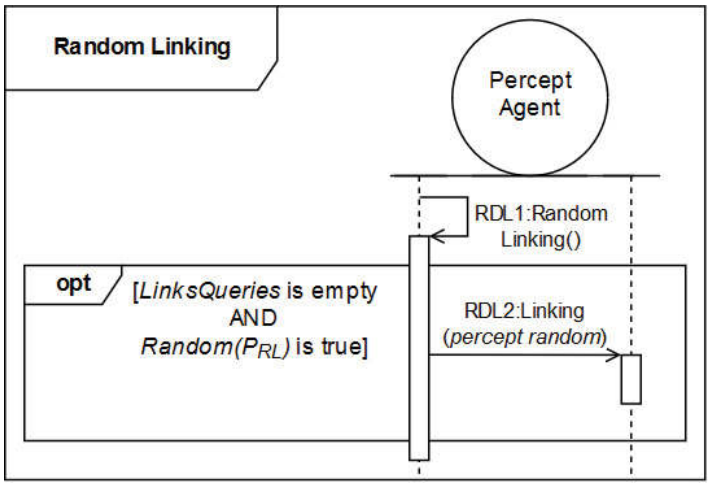

Figure 6.12 — Random Linking Diagram.

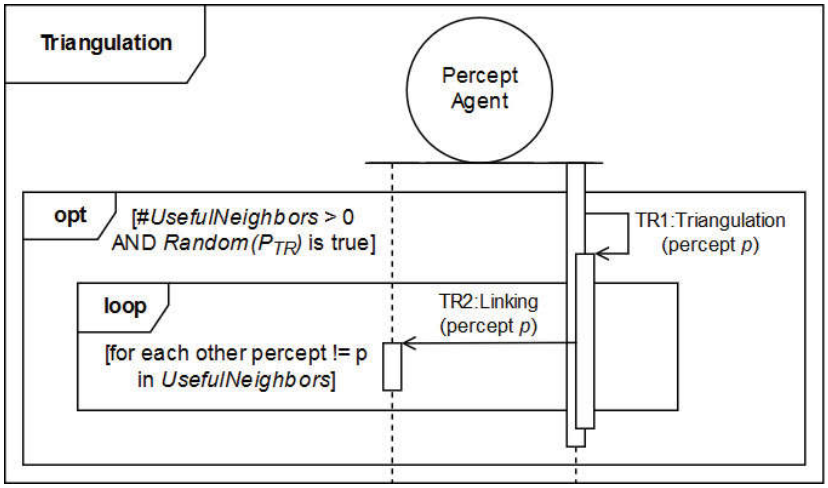

Figure 6.13 — Triangulation Diagram.

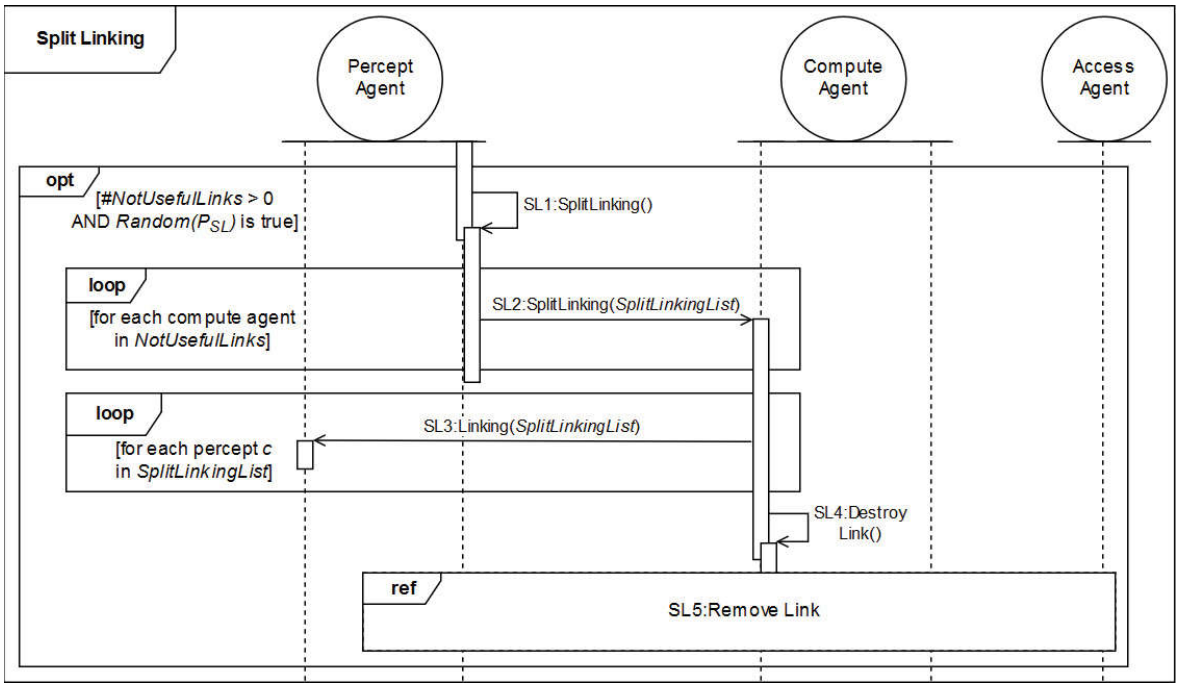

Figure 6.14 — Split Linking Diagram.

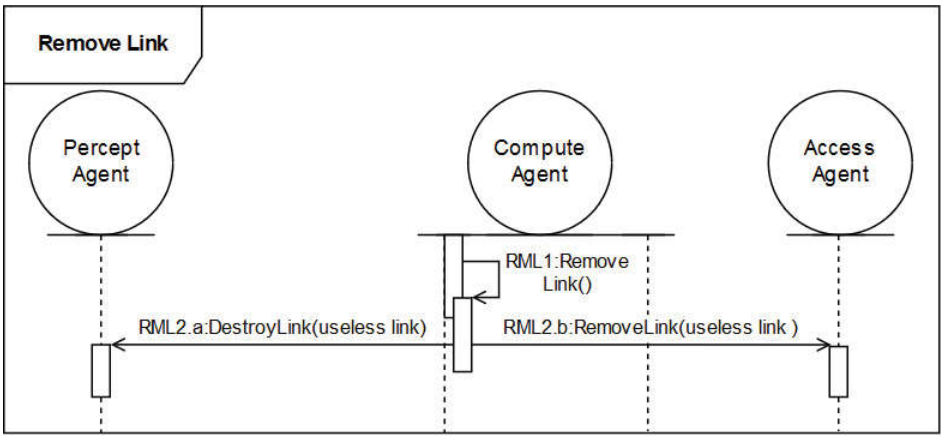

Figure 6.15 — Remove Link Diagram.

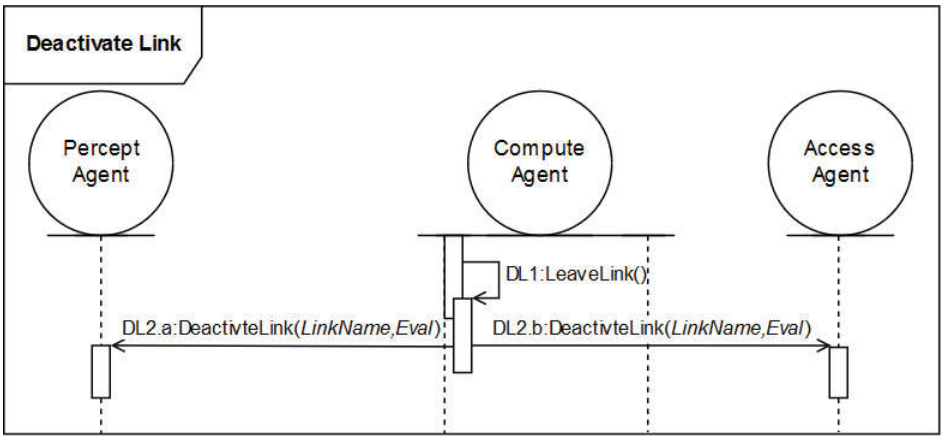

Figure 6.16 — Deactivate Link Diagram.

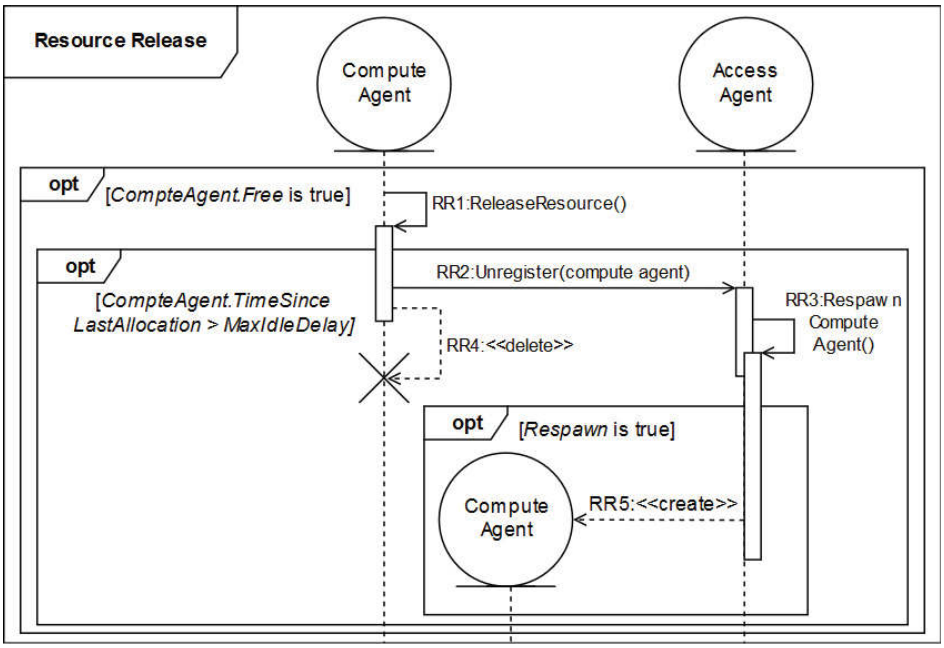

Figure 6.17 — Resource Release Diagram.

7 DREAM: Main Implementation of AMAS4BigData

The goal of this chapter is to present how a user can implement an analytics system with the AMAS4BigData framework though a concrete application called *"DREAM"*[1].

This application builds and displays, in real time from huge amounts of data, a data *model* in form of a dynamic graph, that adjusts itself to adapt to changes in data content and structure. A node represents a data source (sensor stream, database attribute, data file column...) and an edge exhibits a complex correlation between the two related data sources to help the users in finding relevant relations.

The analytics tool DREAM is an implementation of the AMAS4BigData framework for dynamic data relations discovery based on a new data similarity metric, the *Dynamics Correlation*, which is the data transformation method that also contributes to evaluate the usefulness of an analytic link (cf. 6.4. Framework Use).

So I first describe this *Dynamics Correlation* metric and how it is used by the agents to extract relations between the data sources. Additionally, I extend one behavior of the percept agents and extend the criticality function with dedicated indicators. Finally, I briefly present some other possible analytics applications, which are re-implementations of conventional data analytics tool using AMAS4BigData.

[1] *DREAM* stands for *D*ynamic data *R*elation *E*xtraction using *A*daptive *M*ulti-agent systems

7.1 Dynamic Big Data Similarity Metric: Dynamics Correlation

DREAM relies on a new analytical tool, designed from conventional analytical tools, that can transform dynamic big data streams into a compact representation of their dynamics (evolution over time), which exhibits the changes happening in the data streams, and compare how these streams dynamics are similar.

Put it simply, the *Dynamics Correlation* aims to describe how similar is the evolution of two data streams even if they are time shifted.

In the following, the origin and the composition of the *Dynamics Correlation* are given.

7.1.1 Correlation coefficient

When investigating relations between data, one relies first on the most widespread analytical tool, the statistical correlation defined with the Pearson's correlation coefficient r [253] as follows:

$$r(A,B) = \frac{\sum_{i=1}^{n}(A_i B_i) - n\bar{A}\bar{B}}{n\sigma_A \sigma_B} \tag{7.1}$$

Where,

- A, B are two variables (data features),
- X_i is i^{th} value of X, with i starting from 1,
- $\bar{X}$ is the mean of X,
- σ_X is the standard deviation of X,

- n is the number of data points (values).

We use the correlation magnitude r^2 to measure the strength of the correlation : the greater r^2 is, the stronger is the correlation.

However, the correlation coefficient has a major downside: non-linearly correlated variables (data attributes) are considered as independent ($r \approx 0$) but they are in fact correlated and therefore it causes a loss of relevant information. The non-linear correlation is mainly due to:

▷ **Non-linearity:** when at least one of the variables is non-linear or random-like (cf. fig.7.1). In this example, there is a non-linear correlation between the variables but, as the scatter plot[2] shows, the conventional correlation is unable to detect such complex relations.

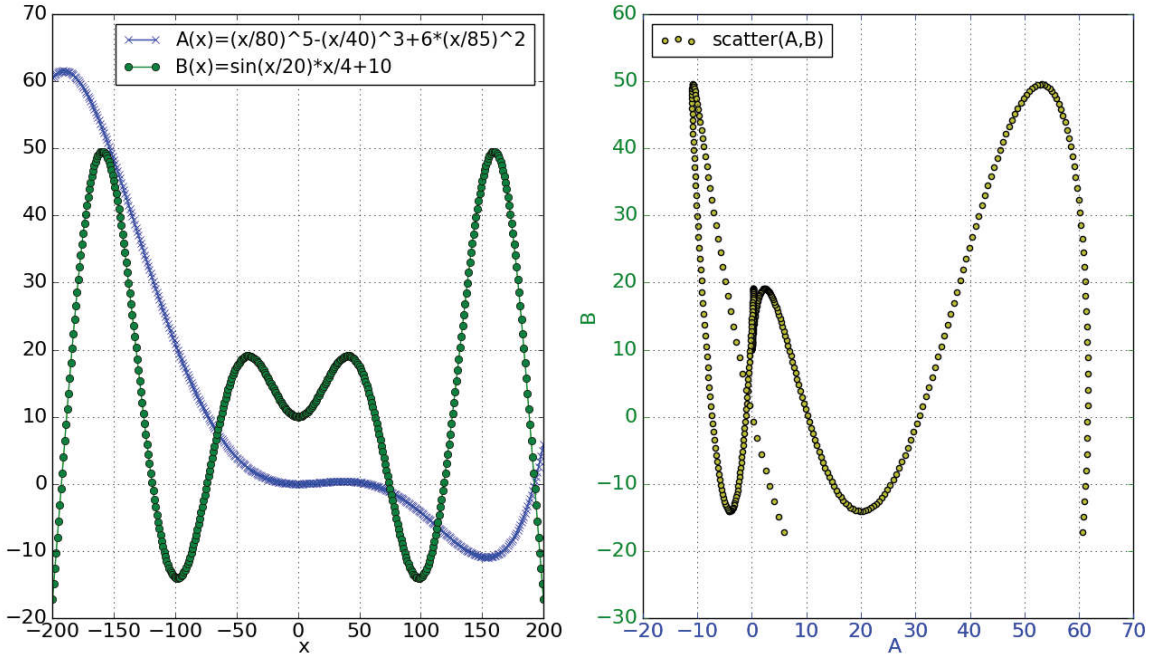

Figure 7.1 — Non-linearity.

▷ **Time shift:** for example, in figure 7.2 $A(x) = sin(x)$ and $B(x) = sin(x + \frac{\pi}{2})$. Both are sinus functions that take the same argument (x), with a delay of $\frac{\pi}{2}$ for the second one ($B(x)$).

This time shift leads to non-linear correlations, also undetected by the correlation coefficient.

▷ **Non-linearity and Time shift:** is the worst case, where the two variables are non-linear and time shifted (cf. fig.7.3), which is the case of many real world data. Thence their correlation is even more complex.

[2] The visual representation of two variables correlation, built by projecting one variable on another.

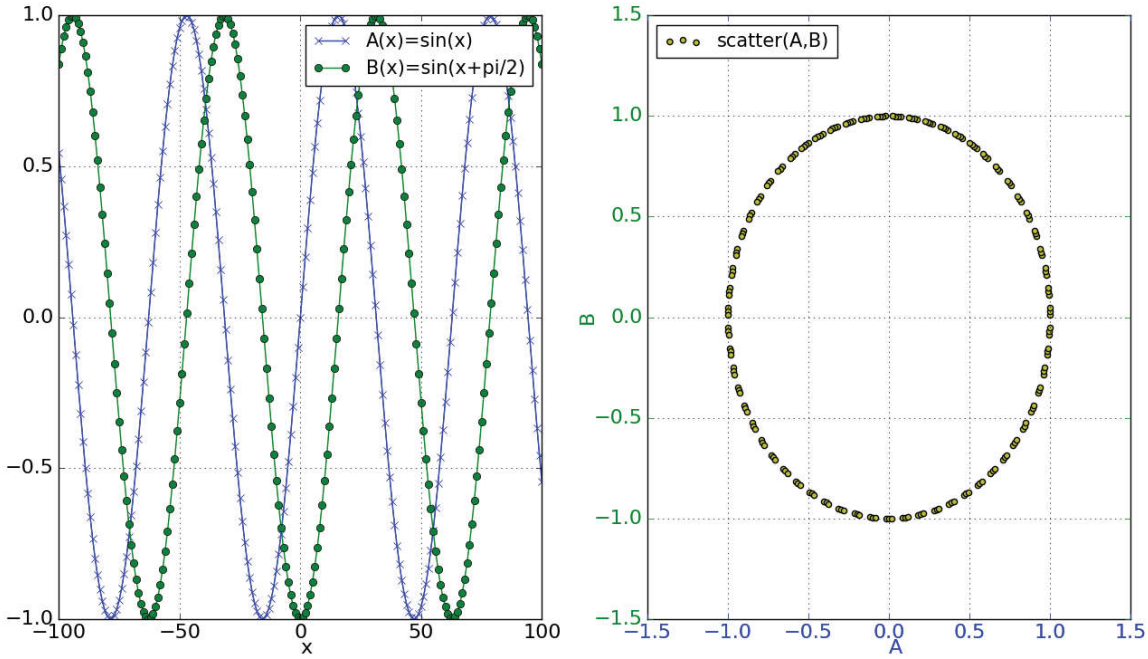

Figure 7.2 — Time shift.

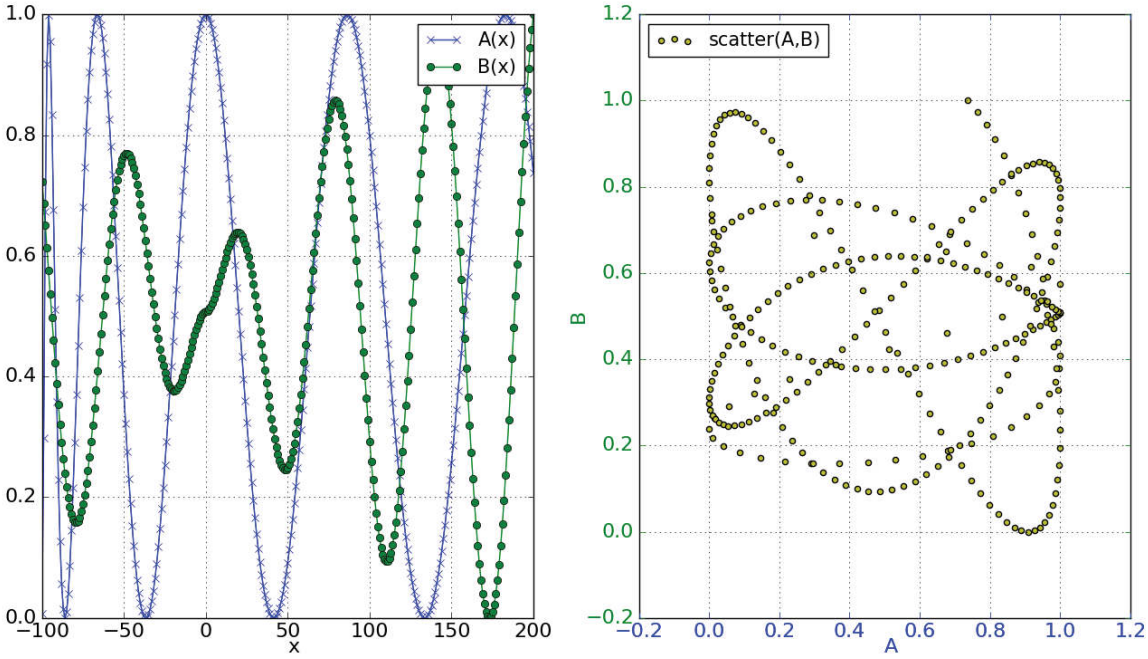

Figure 7.3 — Non-linearity & Time shift.

Therefore, in order to distinguish between a true independence and a non-linear correlation, one can look for a pattern or a distinctive shape in the scatter plot (cf. fig.7.2, fig.7.1 and fig.7.3), which suggests a non-linear correlation. Conversely, a uniform scatter plot indicates a true independence (cf. fig.7.4 right).

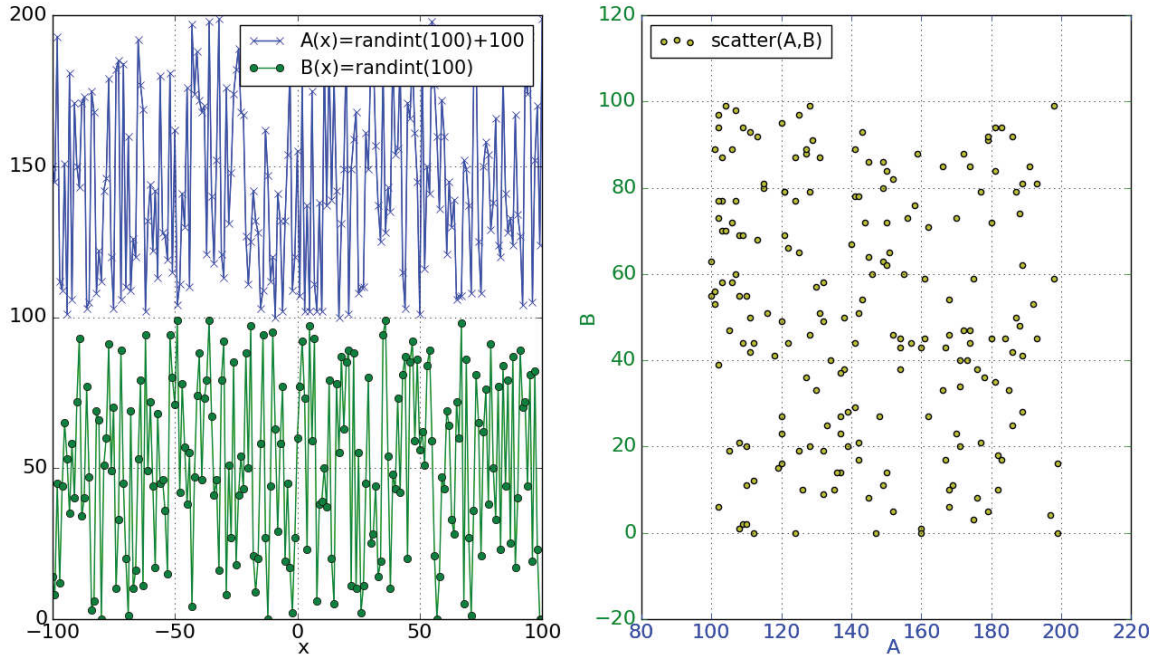

Figure 7.4 — true independence.

Nonetheless, this process can be quiet difficult and non-systematic. Hence another analytical tool based on a similar process is needed.

7.1.2 Phase Space Similarity

The defining feature of a scatter plot is its graphical representation of the relation between two variables over time. This representation may be so complex that it looks like a uniform scatter, which hints at an independence, as a result of the projection of one complex variable over another one.

I exploit another representation of the variables, that avoids the projection, to study the relation between them. This representation, known as *Phase Space*, came from physics[254] and is built following the behavior of a single variable over time, also known as *Dynamics*.

7.1.2.1 Phase Space

Formally, the Phase Space PS is a collection of points, whose coordinates are the difference (or distance) of three successive data points (three data values are required to get one point in the phase space) in a sliding window, defined as follows:

$$(psx_{A_i}, psy_{A_i}) = (A_i - A_{i-1}, A_{i+1} - A_i) \tag{7.2}$$

$$PS_A = \{(psx_{A_i}, psy_{A_i}), \forall i \in [1, n-1]\} \tag{7.3}$$

In other words, a phase space is the projection of the distance of the two previous successive data points (axis of abscissa) on the distance to the next point (ordinate axis).

Thus, it displays the behavior of one variable, or function, with an abstraction of the data temporal dimension. Put it simply, a phase space is the representation of a variable dynamics in a nutshell.

To better understand the concept of phase space, some graphical examples of phase spaces, classified in three dynamics categories, are presented hereafter:

- **Polynomial Dynamics**: the phase space of polynomial function results in a regular and constant pattern. Furthermore, the more simple the dynamics of one variable, the more constant is its phase space.

  This is especially true with a simplistic polynomial functions like the one shown in the first example (cf.fig.7.5). The corresponding phase space is a single point, or more specifically a set of the same exact points, which translates a constant distance between all the data point.

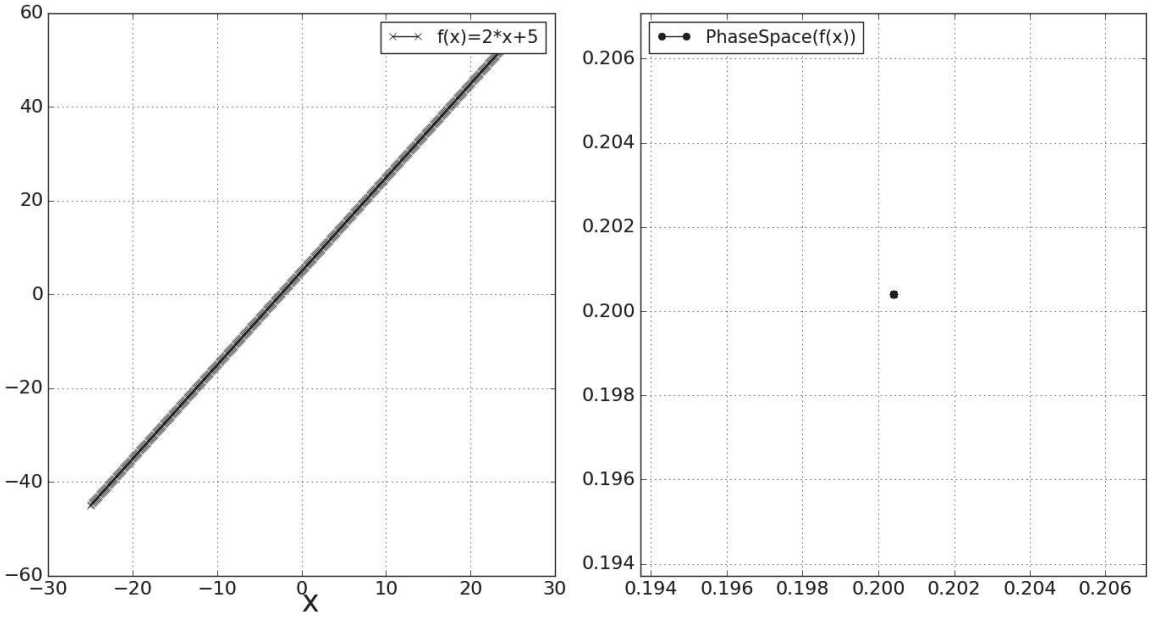

Figure 7.5 — Example1 of polynomial dynamics.

The next examples (cf. fig.7.6, fig.7.7, fig.7.8) show phase spaces with regular patterns, which means that the successive distance between the data points is increasing or decreasing regularly.

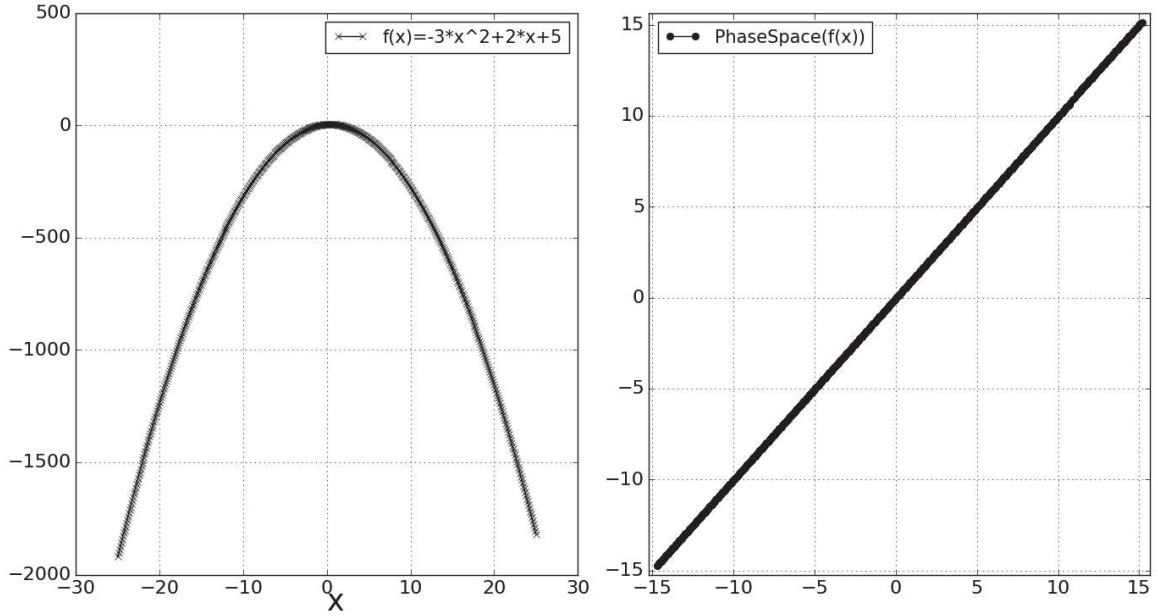

Figure 7.6 — Example2 of polynomial dynamics.

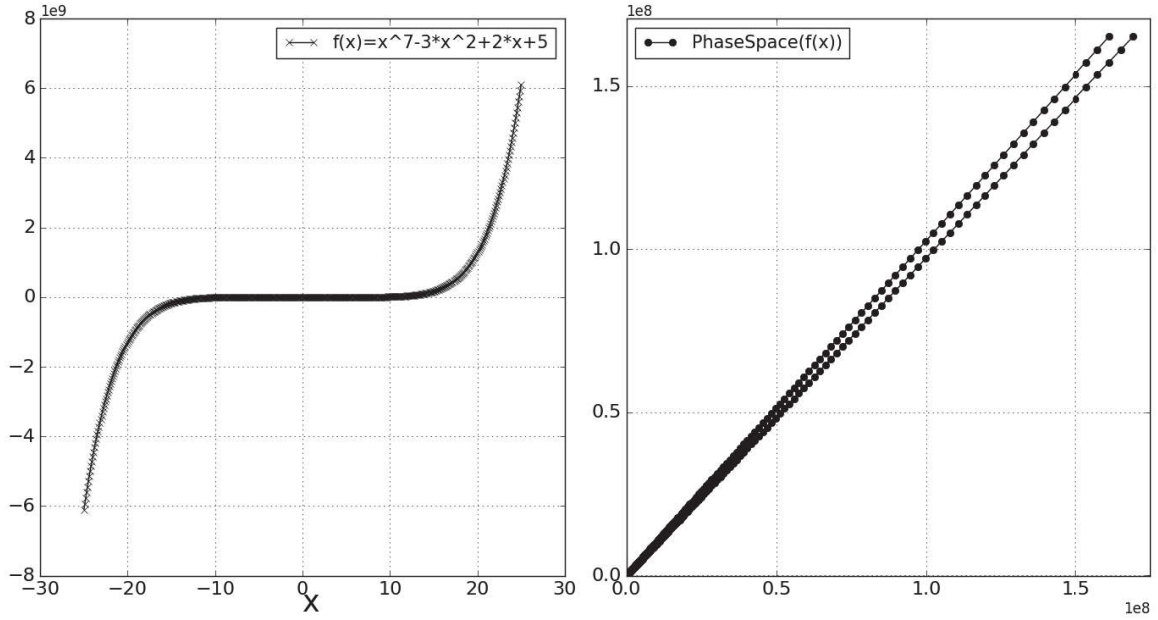

Figure 7.7 — Example3 of polynomial dynamics.

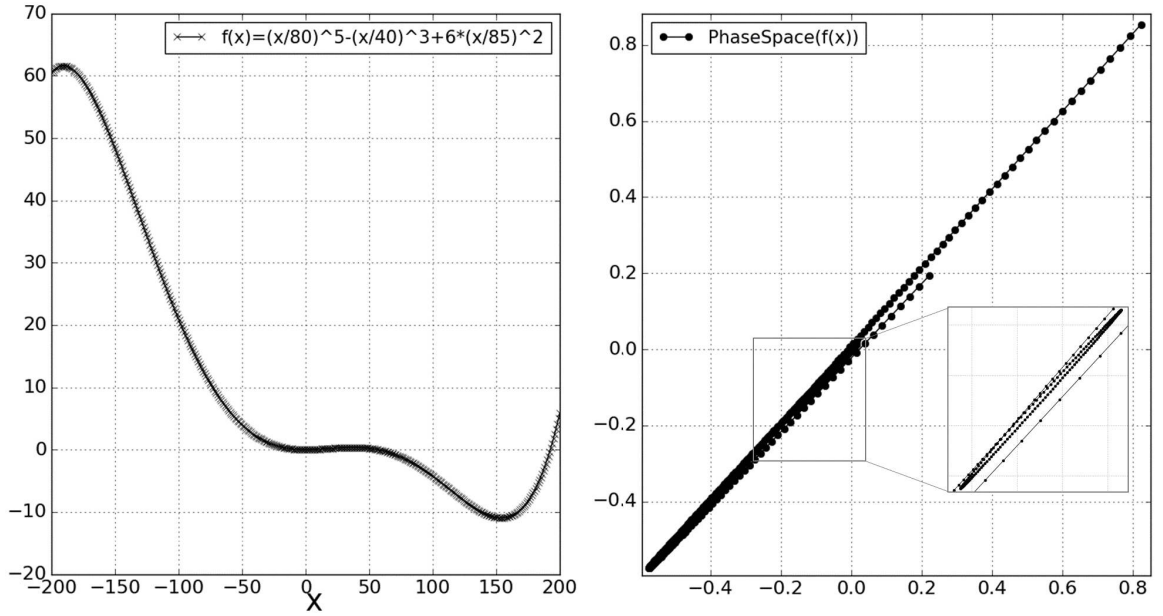

Figure 7.8 — Example4 of polynomial dynamics.

▷ **Cyclic Dynamics**: in this category, the phase space presents a repetitive ellipse-like pattern that translates the periodically repetitive behavior of one function, like the sinusoidal functions given in the following examples (cf.fig.7.9, fig.7.10, fig.7.11).

Similarly to the first dynamics category, the complexity of the phase space pattern of cyclic dynamics grows with the complexity of its origin, the data variable.

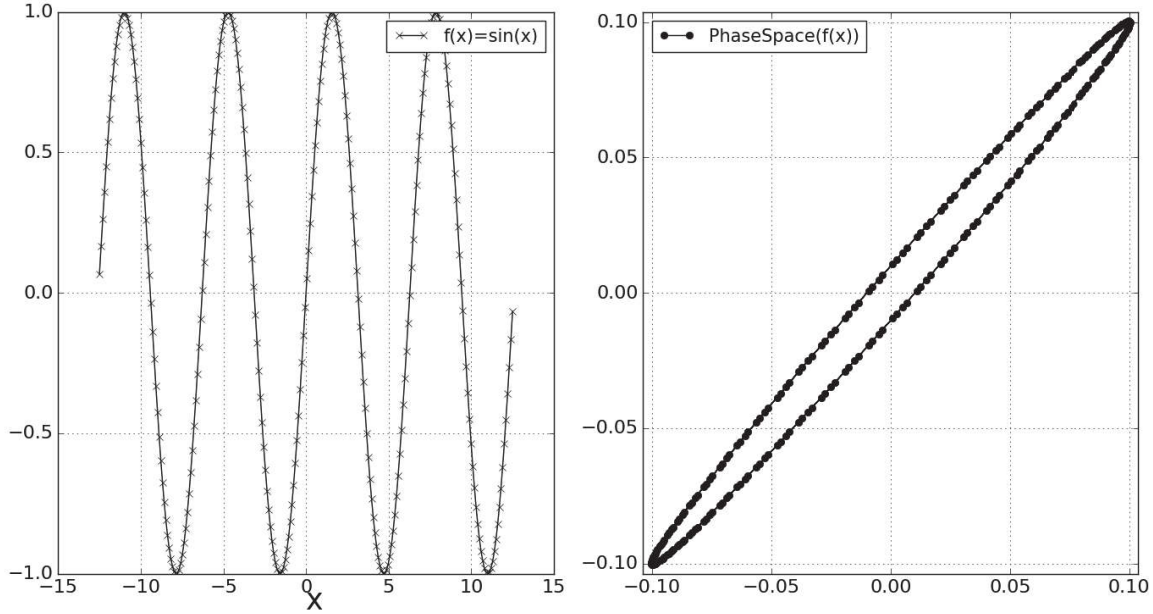

Figure 7.9 — Example1 of cyclic dynamics.

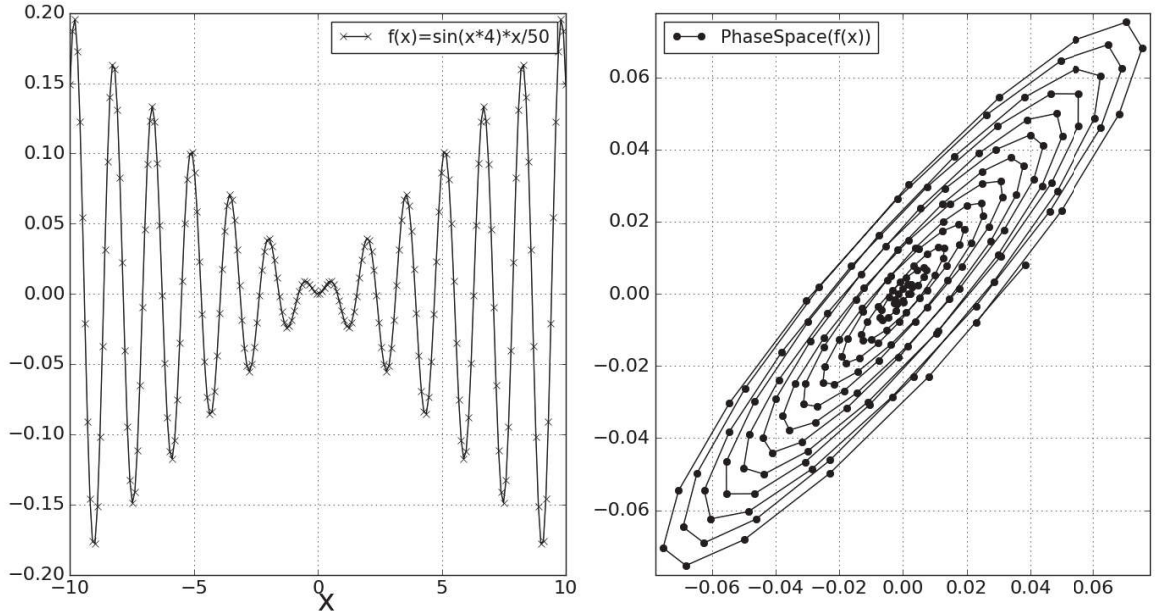

Figure 7.10 — Example2 of cyclic dynamics.

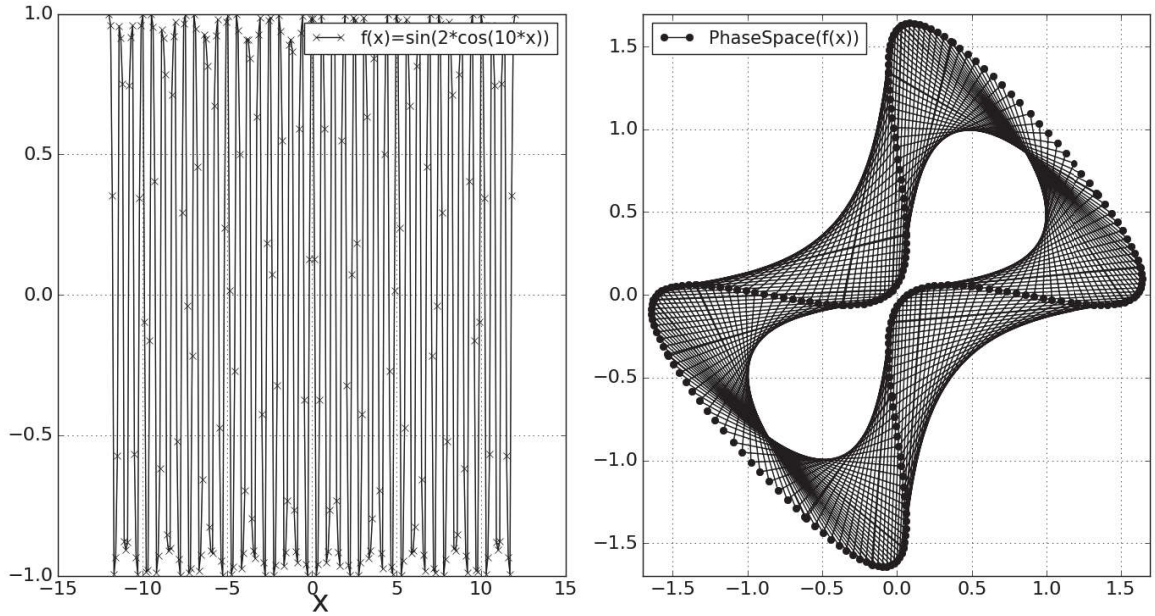

Figure 7.11 — Example3 of cyclic dynamics.

▷ **Random Dynamics**: the last category of dynamics concerns the data variable that are random-like. Therefore, their phase space is chaotic and no distinctive pattern can be found because of the variable randomness, meaning that there is no logical progression of the successive distances as illustrated in figure 7.12.

Yet, even if there is no pattern in a phase space, it remains useful to compare the dynamics of two non-linear variables and/or time shifted variables, because variables with

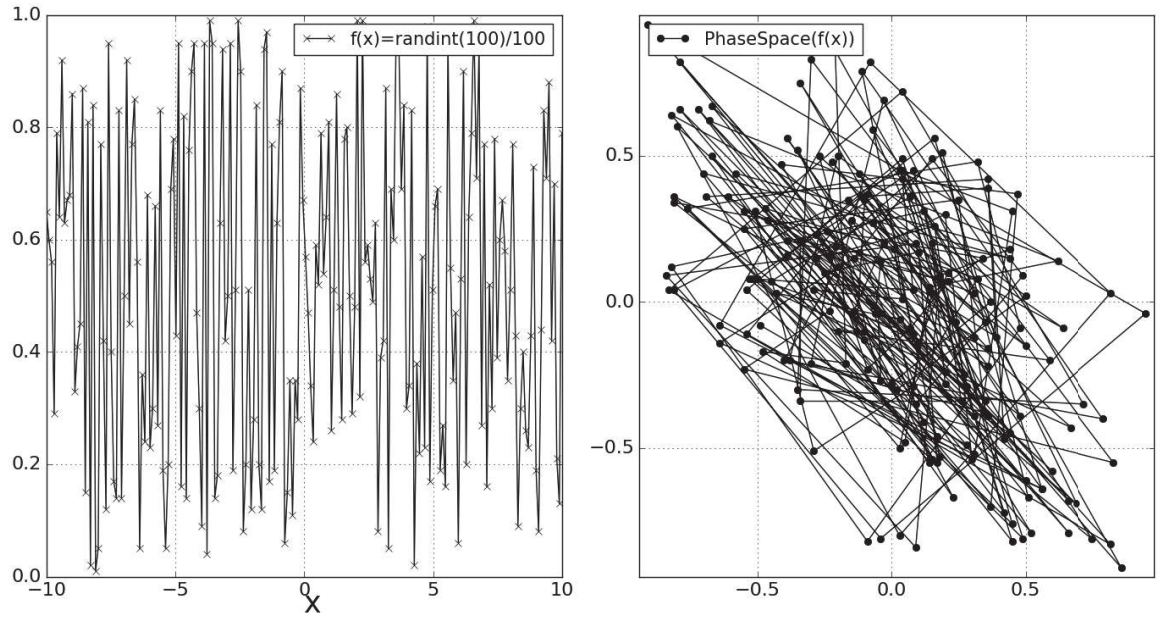

Figure 7.12 — Example of random dynamics.

similar dynamics have a similar phase space (cf.fig.7.13). This phase space similarity is described in the following.

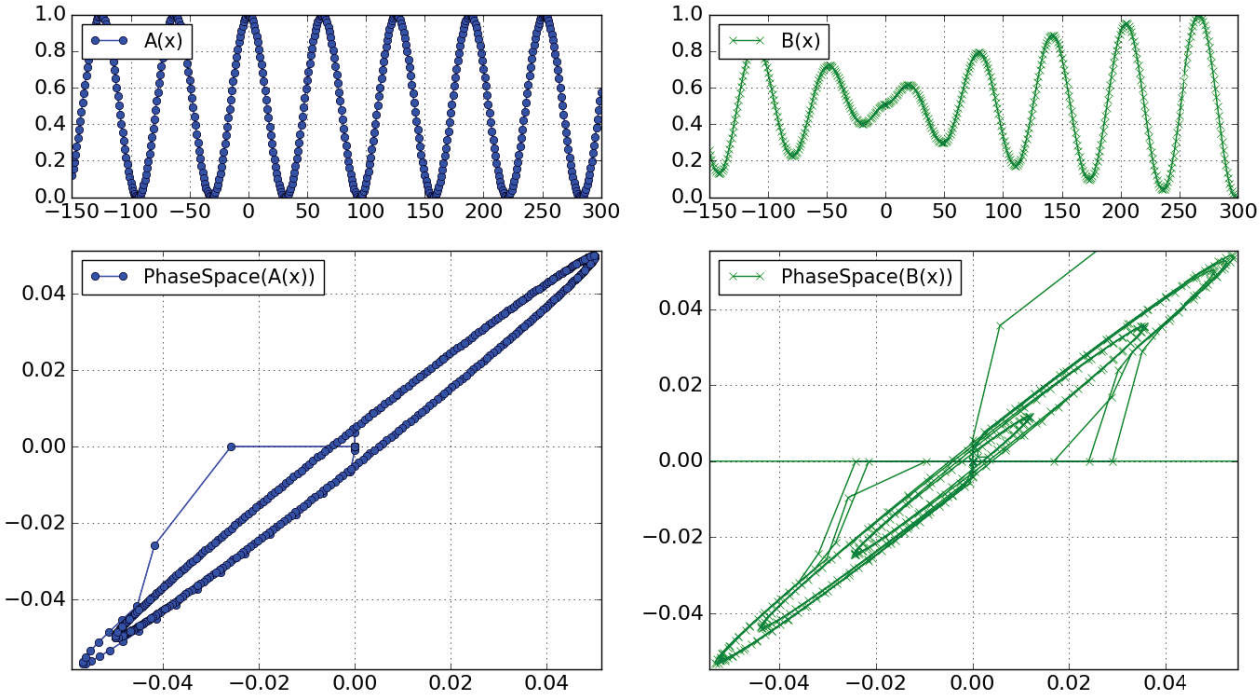

Figure 7.13 — Phase Spaces of similar variables.

7.1.2.2 Phase Space Similarity metric

The *Phase Space Similarity (PSS)* measures how close two phase spaces are, implying how much two variables dynamics are alike.

As a similarity metric the *PSS* should vary from 0 to 1, meaning full dissimilarity and full similarity respectively. Accordingly, I define the *PSS* metric for an automatic comparison between two phase spaces, as follows:

$$PSS(A,B) = \left(1 - PSED(A,B)^2\right)^2 \tag{7.4}$$

$$PSED(A,B) = \frac{\sum_{i=2}^{n-1} \sqrt{\left(psx_{A_i} - psx_{B_i}\right)^2 + \left(psy_{A_i} - psy_{B_i}\right)^2}}{(n-2)\ \ Max\_ED(A,B)} \tag{7.5}$$

$$Max\_ED(A,B) = \begin{cases} \sqrt{2} & \text{if } A \text{ and } B \text{ are normalized} \\ \sqrt{\left(\max\limits_{2\le i\le n-1} psx_i - \min\limits_{2\le i\le n-1} psx_i\right)^2 + \left(\max\limits_{2\le i\le n-1} psy_i - \min\limits_{2\le i\le n-1} psy_i\right)^2} & \text{else} \end{cases} \tag{7.6}$$

Formally, the *PSS* is the squared complement of the squared *Phase Space Euclidean Distance* (*PSED*), which is the mean euclidean distance between the phase spaces points of A and B.

Moreover, if the data are not normalized their phase spaces will have different scales and the *PSS* might be negative, even though they have the same dynamics. Thus, the (*PSED*) in equation 7.5 has to be divided by the *Maximal Euclidean Distance* (*MAX_ED*), formulated in equation 7.6.

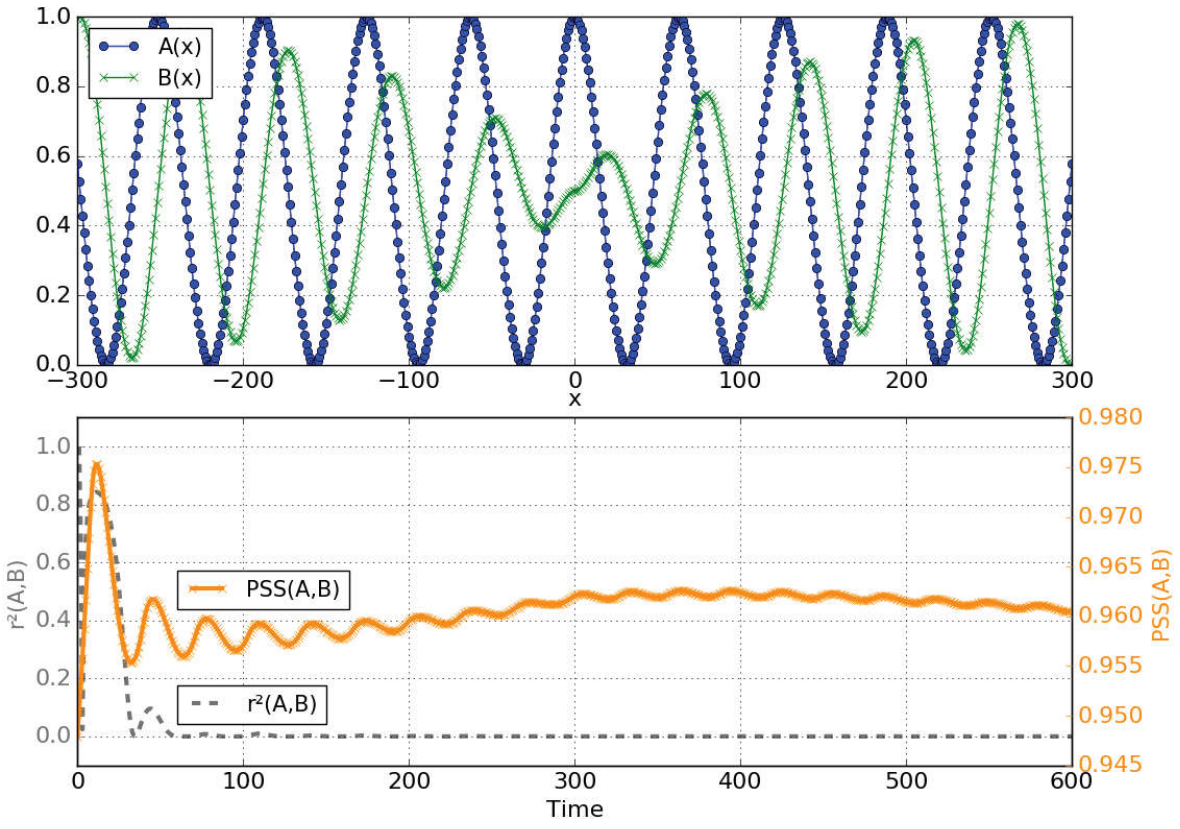

Figure 7.14 — Phase Space Similarity (*PSS*) VS Correlation Coefficient (r^2).

As shown in figure 7.14, the correlation coefficient r^2 starts with high values then quickly decreases and remains very close to 0. Whereas the *PSS* converges towards 0.96, meaning a high similarity between the phase spaces of the two non-linear and time shifted variables A and B, which indicates a non-linear correlation between the variables that the correlation coefficient could not expose.

7.1.2.3 Local Phase Space Similarity

The *PSS*, like the correlation coefficient, uses an arithmetic mean to compute the mean euclidean distance of the phase spaces points from the beginning (cf. equation 7.5). Therefore, it gives an overall similarity value and muffles short *Situations of Interest (SI)*[3] contained in data that are mostly not correlated.

In other words, relevant *situations of interest* may be drowned in the data as a consequence of the *PSS* memory (arithmetic mean). For instance, in figure 7.15 there are two SIs, respectively to the intervals $[100, 150]$ and $[260, 295]$.

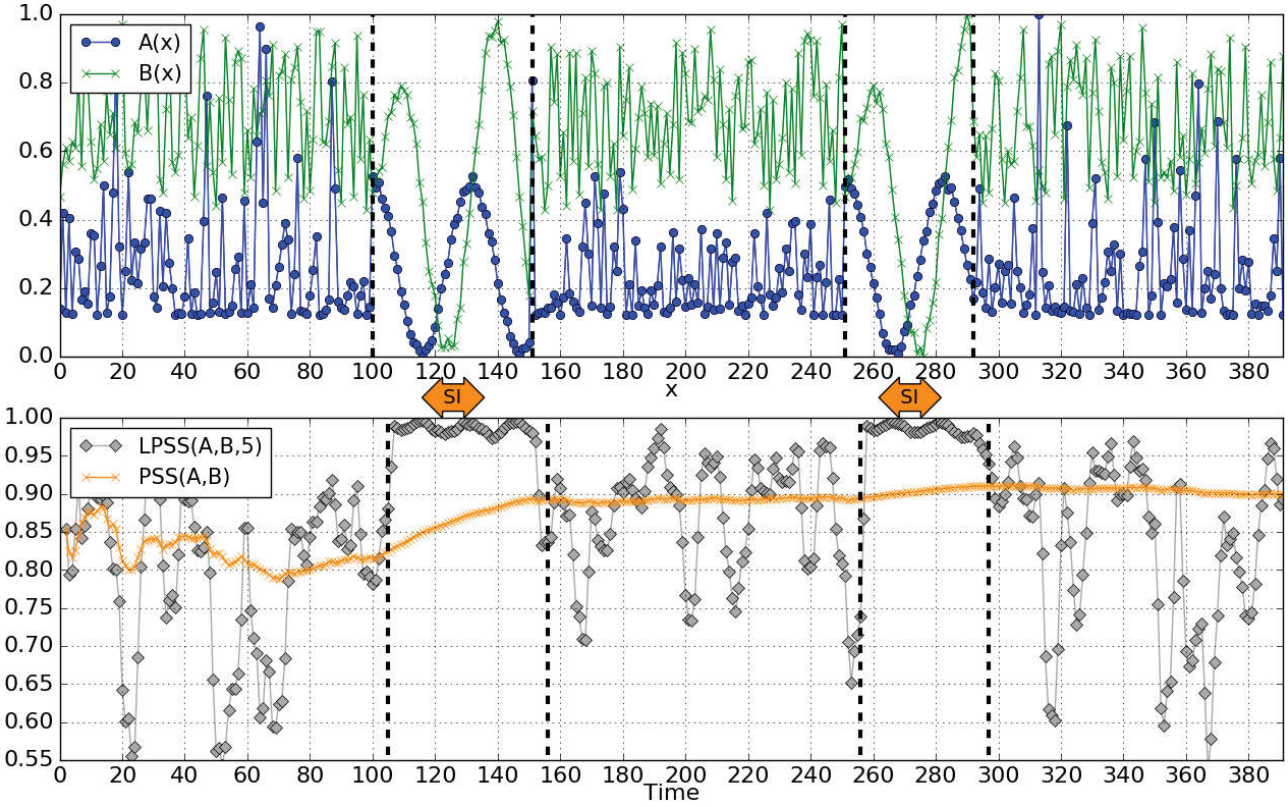

Figure 7.15 — Local Phase Space Similarity (*LPSS*) with $m = 5$.

Therefore, I define the *Local PSS (LPSS)* as a *PSS* with no or very short memory, meaning

[3] Time intervals wherein data (values) might be correlated.

it is a PSS given only the last phase spaces points, expressed as follows:

$$LPSS(A_i, B_i) = \frac{\sum_{j=1}^{m} \left(1 - LPSED(A_{i-j+1}, B_{i-j+1})^2\right)^2}{m}, m \geq 1 \tag{7.7}$$

$$LPSED(A_i, B_i) = \frac{\sqrt{(psx_{A_i} - psx_{B_i})^2 + (psy_{A_i} - psy_{B_i})^2}}{Max\_ED(A, B)} \tag{7.8}$$

Where m is a smoothing factor, or the memory, of the metric output used to lessen occasional disturbances. For example, in figure 7.16, m is set to 5, as a result the $LPSS$ fits the data dynamics well enough while filtering outlier values.

Moreover, the data shown in figure 7.14 have a very high PSS when analyzed as it is. When these correlated data are inserted in random data (cf. fig.7.15) they are still considered as a *situations of interest (SI)*.

On one hand, when the SI starts, the PSS increases slowly although it doesn't reach the very high values, like in figure 7.14, as a consequence of the random data before the SIs that lowers the overall PSS. Then, after the ending of the SIs, the PSS slowly decreases.

On the other hand, the $LPSS$ varies randomly between 0 and 1 for the data before and after the SIs, whilst the $LPSS$ points out very well to the correlated data by varying between 0.96 and 1 during the SIs.

7.1.3 Dynamics Correlation

When one variable is constant-like and the other one is highly dynamic, like respectively A and B during $S2$ in figure 7.16, the phase space of the former overwhelms the phase space of the latter leading phase space based similarity metrics (PSS and $LPSS$) to be falsely high.

To fix this issue of false positives, I define the *Dynamics Correlation* metric, a combination of the $LPSS$ with the r^2 coefficient of the data in the SI ($r(A_{SI}, B_{SI})^2$), as follows:

- If $LPSS(A_{SI}, B_{SI}) \geq 0.95$ AND $r(A_{SI}, B_{SI})^2 \geq 0.05$ it is a true situation of interest ($S1$).
- If $LPSS(A_{SI}, B_{SI}) \geq 0.95$ AND $r(A_{SI}, B_{SI})^2 < 0.05$ it is a false situation of interest ($S2$).
- If $LPSS(A_{SI}, B_{SI}) < 0.95$ it is not a situation of interest ($S3$).

with,

$$LPSS(A_{SI}, B_{SI}) = \frac{\sum_{i \in SI} LPSS(A_i, B_i)}{\#SI} \tag{7.9}$$

$$r(A_{SI}, B_{SI}) = \frac{\sum_{i \in SI}(A_i B_i) - n\bar{A}\bar{B}}{n\sigma_A \sigma_B} \tag{7.10}$$

Where the mean $\bar{A}$ and the standard deviation σ_A are updated incrementally [255] as follow:

$$\bar{A}_i = \frac{U_i}{i} \qquad with \quad U_i = U_{i-1} + A_i, \quad U_0 = 0 \tag{7.11}$$

$$\sigma_{A_i} = \frac{V_i}{i} - \bar{A}_i^2 \;\; with \;\; V_i = V_{i-1} + A_i^2, \;\; V_0 = 0 \tag{7.12}$$

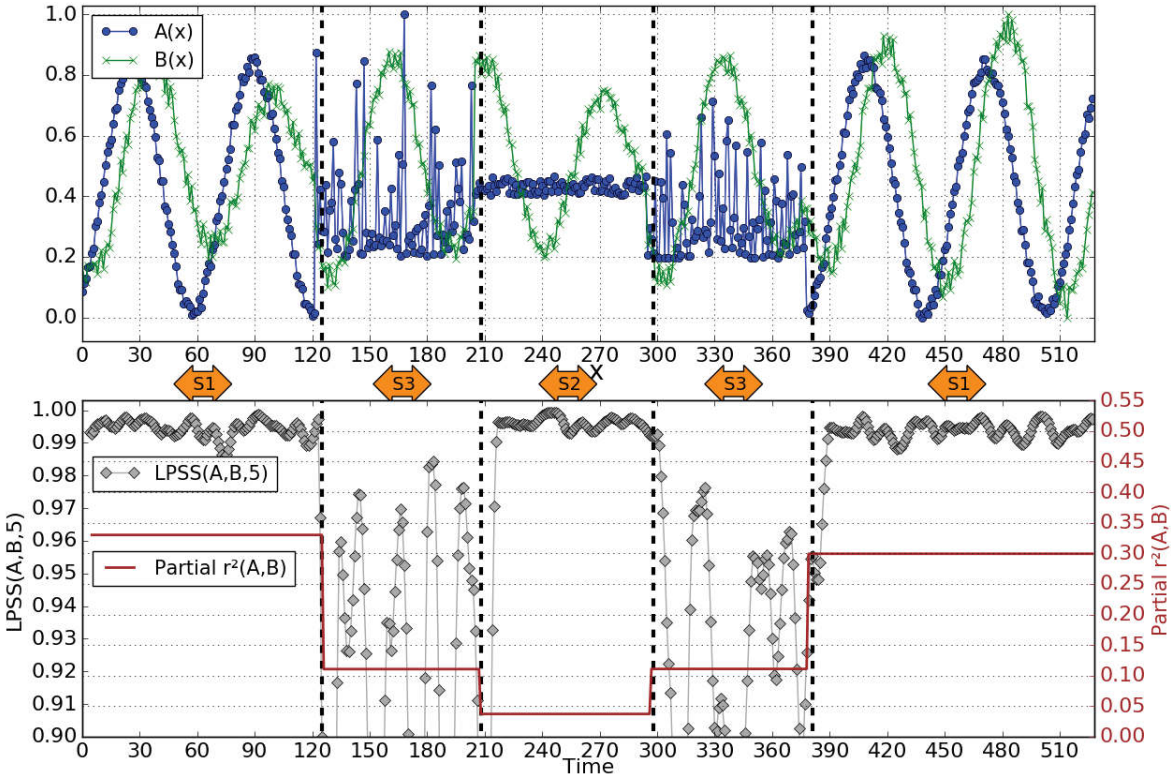

Figure 7.16 — Dynamics Correlation. with artificial noise in *S2* and *S3*.

In addition, the strength of the dynamics correlation (DC) is evaluated incrementally as:

$$DC(A,B) = \frac{DC(A,B) + DC(A_{SI},B_{SI}) * \#SI}{\sum_{SI} \#SI}, \text{ with initially } DC(A,B) = 0 \tag{7.13}$$

$$DC(A_{SI},B_{SI}) = Max(LPSS(A_{SI},B_{SI})^2, r(A_{SI},B_{SI})^2) \tag{7.14}$$

7.2 DREAM Architecture: Implementation of AMAS4BigData

Since DREAM is an application of the AMASBigData framework, it is composed of percept agents, compute agents and an access agent, with their corresponding attributes, behaviors, and criticality function.

In this section I describe the implementation of AMAS4BigData to obtain DREAM, as explained in 6.4., by specifying the pre-processing method of the percept agents, the compute agents data transformation method and the usefulness evaluation function of the analytical links, and the percept agents information processing method. In addition, I extend one of the percept agents behaviors with a new perception and the criticality function with some extra indicators.

But first of all, I give a broad overview of DREAM, from the analytics process view, that shows how all the previous elements fit together.

7.2.1 DREAM Analytics Process Overview

DREAM is an analytics system that aims to find relations between data and as described in the framework functioning (cf. 6.3): first the percept agents pre-process the new raw data, the compute agents transform the pre-processed data into information, then from the gathered information the percepts build and update the global knowledge locally.

Algorithm 7.1: DREAM Analytics Process

Input : the raw *data* to be processed
Output: Data graph model /* the combined data *dynamics relations* */
1. **forall the** *raw data* **do**
2. $p \leftarrow data.Percept$; /* p is the percept agent corresponding to the source of *data* */
3. $p.$***IncrementalNormalization***$(data)$; /* send the *data* to its corresponding percept agent p so it pre-processes the *data* with *algorithm* 7.2 */
4. **forall the** *Compute Agents c in p.DataConsumers* **do**
5. $c.$***DataTransormation***$(p.normalized\_data)$; /* send the *normalized_data* to the associated compute agents so they transform them into information following *algorithm* 7.3 */
6. $c.$***LinkEvaluation***$()$; /* Evaluate the usefulness of the link and follow the procedure of its life cycle (cf. 6.1) */
7. $c.firstPercept.$***InformationProcessing***$()$; /* send the information produced to the associated percepts, so they can process it */
8. $c.secondPercept.$***InformationProcessing***$()$;

7.2.2 Raw Data Pre-Processing: Normalization

The pre-processing of the raw data acquired by DREAM is an incremental min-max normalization made by the percept agents, each time a new data is received, with algorithm 7.2:

Algorithm 7.2: PerceptAgent.IncrementalNormalization()

Input: the *raw_data* to be pre-processed
1. $min\_value \leftarrow Min(min\_value, raw\_data)$
2. $max\_value \leftarrow Max(max\_value, raw\_data)$
3. $normalized\_data \leftarrow \frac{raw\_data - min\_value}{max\_value - min\_value}$

This pre-processing allows all the data to be on the same scale ($[0,1]$) which avoids errors in the analytical tools. Once the raw data is pre-processed by the percept agents, it is sent to the related compute agents, which transform it into information.

7.2.3 Data Transformation: Dynamics Relations Detection

The data transformation, whose goal is to reveal *dynamics relations* between data (true *Situations of Interests* wherein the data dynamics are correlated), is carried out by the compute agent when it receives a data from one of its two related percept agents A and B, by application of the dynamics correlation analytical tool (cf. 7.1.3) following algorithm 7.3.

Algorithm 7.3: ComputeAgent.DataTransformation()

Input: the *data received* one associated percept
A, the first associated percept
B, the second associated percept

1. $data\_buffer[data.Percept].add(received\_data)$; /* store the *received_data* into the *data_buffer* of the sender percept */
2. **if** $data\_buffer[A] \neq \emptyset$ *AND* $data\_buffer[B] \neq \emptyset$ /* if the *data_buffers* of both associated percepts are not empty */
3. **then**
4. $\quad data\_points[A].add(mean(data\_buffer[A]))$; /* add the mean value of the *data_buffer* to the *data_points* list in order to balance the difference between the data generation frequencies of the percepts */
5. $\quad data\_points[B].add(mean(data\_buffer[B]))$;
6. $\quad data\_buffer[A] \leftarrow \emptyset$; /* empty the *data_buffer* to store future data */
7. $\quad data\_buffer[B] \leftarrow \emptyset$;
8. **if** $\#data\_points[A] \geq 3$ *OR* $\#data\_points[B] \geq 3$ /* there is enough *data_points*, at least 3, of each percept to produce information */
9. **then**
10. $\quad$ ***InformationProduction***$()$; /* use the *Dynamics Correlation* analytical tool (cf. 7.1.3) to produce information (cf. algo.7.4) */

Note that initially, *data_buffer* and *data_points* are empty, IN_SI and $TRUE\_SI$ are false, and $TimeSinceLastSI$, which is used later by the *LinkEvaluation* function (cf. 7.2.4), is set to 0 and incremented while no SI (*Situation of Interest*) is detected.

To sum up, this algorithm gives the steps that a compute agent follows to continuously transform the normalized data, received from its percepts, into information. In this case, information means the true SIs and their strength ($DC(A, B)$).

These steps can be summarized as: first store the received data in a data buffer (temporary storage) dedicated to one percept. Then once there is data in both data buffers, take their mean values, use them to extend the corresponding phase spaces and update their similarity ($LPSS$) in order to find SIs.

Finally, to extract a dynamics relation between the two percepts data, the agent must distinguish between true and false SIs thanks to their length and partial correlation ($r(A_{SI}, B_{SI})^2$), by means of the sub-procedure *DetectDynamicsRelation()* (cf. algo.7.5).

The *min_len* is the minimal length that all SIs must have in order to consider them as true

Algorithm 7.4: ComputeAgent.InformationProduction()

Input: A, the first associated percept
B, the second associated percept
i, the current analytics cycle (step)

1. **UpdatePhaseSpace**$(A, data\_points[A])$; /* compute a new point of the *PhaseSpace* of A with the three *data_points* of A with formula 7.2 */
2. **UpdatePhaseSpace**$(B, data\_points[B])$;
3. $lpss \leftarrow$ **LPSS**(A_i, B_i); /* compute the Local Phase Space Similarity following the formula 7.7 */
4. **if** $lpss \geq 0.95$ /* the Local Phase Spaces of A and B are Similar, so it is a Situation of Interest (*SI*), meaning the data of A and B are most likely correlated and there is a dynamics relation between them */
5. **then**
6. **if** *not* IN_SI /* if not already in an *SI*, it is the beginning of a new one */
7. **then**
8. $IN\_SI \leftarrow true$;
9. $r \leftarrow \mathbf{r}(A_{SI}, B_{SI})$; /* update the correlation coefficient with data of this SI following equation 7.10) */
10. **if** *not* $TRUE\_SI$ /* this *SI* is not confirmed, there is no dynamics relation */
11. **then**
12. ***DetectDynamicsRelation()***; /* try to confirm the SI, searching for a dynamics relation (cf. Algo 7.5) */
13. **else**
14. **UpdateDC**(A, B); /* the SI is confirmed, then update the dynamics relation strength (cf. eq.7.13) */
15. **else if** IN_SI /* it is the ending of the *SI* */
16. **then**
17. $IN\_SI \leftarrow false$;
18. ***DetectDynamicsRelation()***; /* after the loss of the SI, either confirm or deny the existance of a dynamics relation between the percepts data (cf. Algo 7.5) */
19. $data\_points[A].removeFirst()$; /* remove the first data from the data *data_points* in order to keep 3 data for the next cycle (step) */
20. $data\_points[B].removeFirst()$;
21. $TimeSinceLastSI \leftarrow TimeSinceLastSI + 1$;

SIs. Below this minimal length, true SIs might be skewed due to coincidental high dynamics similarity. This parameter, as other DREAM parameters, is specified by the end users of DREAM and has a default value ($min\_len = 20$).

Algorithm 7.5: DetectDynamicsRelation()

```
1. if #SI ≥ min_len + RandomVal(5) AND r² ≥ 0.05  /* it is a true SI, so there
   is a dynamics relation between the percepts data */
2. then
3.     TRUE_SI ← true;
4.     Count_SI ← Count_SI + 1;
5.     Dynamics_Relation ← true;
6.     UpdateDC(A, B);  /* Evaluate the dynamics correlation strength with
       equation 7.13) */
7. else if TRUE_SI     /* if there is already a Situation of Interest, then
   deny it */
8. then
9.     TRUE_SI ← false;
```

Moreover, a random number between 0 and 5 is added to this minimal length, in order to avoid having all the compute agents detecting dynamics relations at the same time. This would lead to the concurrent execution of the *ValidateLink* behavior and consequently trigger the corresponding behaviors of the related percepts (cf. 6.3.5.). The aim is to further reduce further the potential bottlenecks in the percept agents.

7.2.4 Links Life Cycle & Evaluation

The usefulness of an analytic link is evaluated with the links life cycle algorithm (cf. algo.6.1) by the associated compute agent. In order to do so, the following elements have to be defined:

- **Algorithm Parameters**: for a default use of DREAM, $LinksRemovalTime = 50 + RandomVal(10)$, $LinksDeactivationTime = 30 + RandomVal(5)$, $min\_eval = 0$, however they can be overridden by the end user. As formerly explained, the added random values are used to avoid concurrent links deactivation/removal to reduce the potential bottleneck;

- **LinkEvaluation Function**: tells if an analytical link is useful in terms of information production, meaning true SIs detection, is defined with algorithm 7.6. When the data dynamics are correlated and there is currently an SI, the function returns the strength of the dynamics relation ($DC(A, B)$).

  Furthermore, the function returns -1 when the link has produced no information (SI) at all, or after a given amount of time ($TimeSinceLastSI$) if there is already a dynamics relation, in order to either change the state of the link to *weak* if it was *strong* or to destroy the link if it was *weak* and $RemovalStep$ is reached (cf. algo.6.1).

Algorithm 7.6: LinkEvaluation()

```
1.  if Dynamics_Relation then
2.      if TimeSinceLastSI ≥ Max(Log(∑_SI #SI / 2) + 1), 3 * Log(Count_SI + 2) + 15+
        RandomVal(15)) /* if the dynamics relation is not sustained, no new
        SI for a long time, then remove the relation */
3.      then
4.          TimeSinceLastSI ← 0;
5.          Dynamics_Relation ← false;      /* the dynamics relations is lost */
6.          return -1;                              /* the link becomes weak */
7.      else
8.          return DC(A, B);  /* the link usefulness evaluation is the same as
            the dynamics relation strength */
9.  else
10.     return -1;                                      /* the link is weak */
```

7.2.5 Information Processing: Graph Model Building

DREAM aims to produce a data relations model, which is represented by a dynamic graph wherein a node expresses a data stream (percept) and an edge exhibits a dynamics correlation between two data sources.

Therefore, the percept agents gather the information (dynamics relation) from their related analytical links in order to locally build and update the data model (the graph), according to the rules defined in algorithm 7.7.

Algorithm 7.7: PerceptAgent.InformationProcessing()

1. When a new percept agent is spawned, it creates a new node labeled with the corresponding data source name in the data graph;
2. When a percept (A) links itself with another one (B),
3. the percept with the lowest *ConnectionDegree*[4]becomes the owner of the link;
4. graphically, the links node A to node B with a weak edge, like a dotted line, labeled *"A-B"*;
5. percept A and percept B increase the size of their nodes;
6. Once a *Situation of interest*[5]*(SI)* is detected, the owner changes the state of the link (the color of the edge);
7. At the end of the *Situation of interest*, **if** *a dynamics relation (true SI) is detected between the percept A and the percept B* **then**
8. the owner of the link changes it to a strong one, it changes the corresponding edge as a plain line;
9. and it reflects the strength of the relation ($DC(A, B)$) on the edge, with a color map or opacity level;

10. **else**
11. the owner changes the state of edge back to what it initially was;
12. Also, each time the dynamics relation strength changes, the link owner updates the strength of the link (edge) accordingly;
13. When a dynamics relation is lost, the owner changes the link back to a weak one;
14. When an analytical link is destroyed,
15. the owner removes the corresponding link from the graph model;
16. the size of the nodes previously linked, with the removed edge, decreases;
17. Finally when a percept destroys itself, it removes its node from the graph model;

Hereafter is presented an example of such data graph model (cf. fig.7.17), wherein 22 percepts and the links between them are shown.

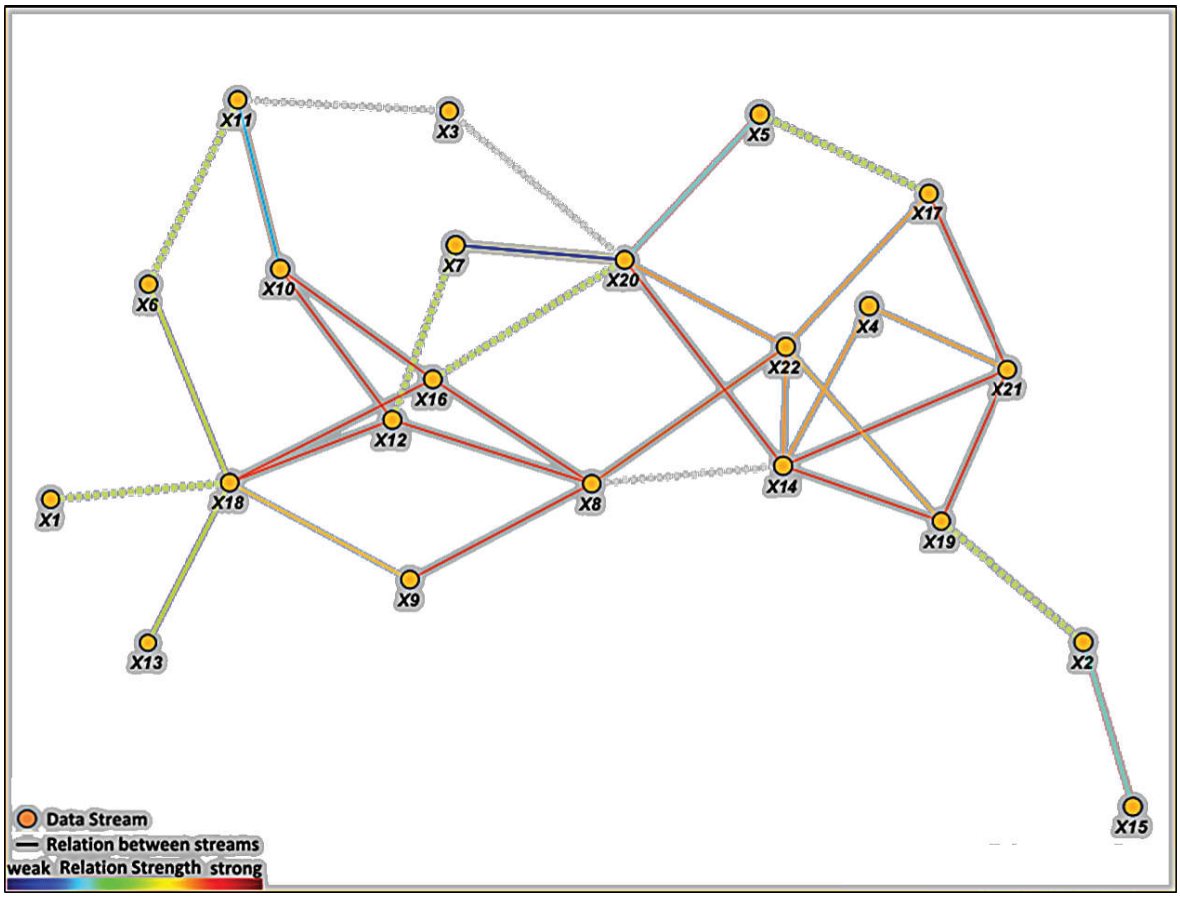

Figure 7.17 — Example of a data graph model.

In this example, some of the edges, the dotted white line like *"X3-X20"*, represent new simple analytical links, and some edges, the green dotted lines as *"X16-X20"*, display analytical links that are in a situation of interest (SI).

[5]The number of links of a percept.
[5]Time intervals wherein the data might be correlated (cf. 7.1.2.3).

The rest of the edges (the plain ones) expresses dynamics relation whose strength is indicated with the color of edge, following a color heat map such as weak dynamics relations tends to blue and strong ones tends to red.

7.2.6 Framework Extensions

As described so far, DREAM implements all the required elements of the AMAS4BigData framework. Thus, it can be used as it is. However, two extensions are made to further improve the exploration of the data space.

7.2.6.1 Random Linking Extension

The proactive exploration behavior *"Random Linking"* (cf. 6.3.2.2) is extended with an additional perception (trigger condition) that drives the percept agents to apply this behavior if they do not find new dynamics relations, thence reducing the stagnation of the relations discovery.

This new trigger condition, which is tested at the end of every analytics cycle, is expressed as follows:

$$Time\_Since\_Last\_Relation \geq Waiting\_Time + RandomVal(5)\ AND\ hasLowLocalDegree()$$

- $Time\_Since\_Last\_Relation$: is an extra attribute of the percept agents, which is a counter similar to $TimeSinceLastSI$ counter. It gives the amount of time (number of analytics cycles) spent by a percept agent since the last relation it discovered;
- $Waiting\_Time$: is the minimal waiting time to trigger the *"Random Linking"* behavior. This waiting duration grows over time, thence preventing useless percepts to consume periodically a computing resource in vain;
- $RandomVal(5)$: alike in $DetectDynamicsRelation()$ function (cf. algo.7.5), this random value prevents the precept agents, that satisfies this condition, to execute the *"Random Linking"* behavior at the same time and thus averting, in limited computing resources systems, the starvation of more critical percept in the near future;
- $hasLowLocalDegree()$: is a function, described in algorithm 7.8, that returns *true* if the percept has a $ConnectionDegree$ lower than the mean of its neighbors $ConnectionDegree$.

Algorithm 7.8: PerceptAgent.hasLowLocalDegree()

1. **if** $\#Neighbors \leq 3$ **then**
2. ⌊ **return** true;
3. $mean\_connection\_degree \leftarrow \frac{\sum_{neighbor \in Neighbors} ConnectionDegree(neighbor)}{\#Neighbors}$;
4. **if** $mean\_connection\_degree \leq 3\ OR\ \#Neighbors \leq mean\_connection\_degree$ **then**
5. ⌊ **return** true;
6. **return** false;

7.2.6.2 Criticality Function Extension

For the sake of a better computing resources allocation, the basic criticality function of the AMAS4BigData framework (cf. 6.3.4.) is expanded with three indicators given hereafter:

1. **Dynamics Correlations Mean (I_1)**: gives an advantage to the percepts with weak mean value of their dynamics relations strength (dynamics correlation), so they can get more computing resources in order to search for new relations, following this:

$$I_1^P = sigmoid\left(1 - \frac{\sum_{c \in DynamicsRelations(p)} DC(p,c)}{\#DynamicsRelations(p)}\right) \tag{7.15}$$

   Where,

   - ▷ $DynamicsRelations(p)$ is a subset of $Neighbors(p)$, wherein each neighbor c has a dynamics relation with p.
   - ▷ $DC(p,c)$ is the dynamics correlation strength (cf. 7.1.3.) of the relation between p and c.

2. **Active Dynamics Correlations Variation (I_2)**: promotes the percept agents whose active dynamics relations[6] have been subjected to a given amount of change in their strength ($Variation\_Threshold$ in equation 7.17), so they can activate their inactive dynamics relations and update them. This indicator is computed with:

$$I_2^P = Log\left(\frac{\sum_{c \in ActiveDynamicsRelations(p)} DC\_Variation(p,c)}{\#ActiveDynamicsRelations(p)}\right) \tag{7.16}$$

$$DC\_Variation(p,c) = \begin{cases} 1 \text{ if } \frac{Min(DC(p,c)_t, DC(p,c)_{t-1})}{Max(DC(p,c)_t, DC(p,c)_{t-1})} \geq Variation\_Threshold \\ 0 \text{ else} \end{cases} \tag{7.17}$$

3. **Dynamics Relations Speed Change (I_3)**: translates the need to allocate more computing resources to the percepts that gain and lose dynamics relations the fastest.

   So this indicator, which expresses the mean speed of the changes occurring on one percept's dynamics relations, and is computed incrementally with:

$$I_3^p(t) = \frac{I_3^p(t-1) + DRC(p,t)}{2} \tag{7.18}$$

$$DRC(p,t) = \frac{|\#DynalicsRelations(p)_t - \#DynalicsRelations(p)_{t-1}|}{Max(\#DynalicsRelations(p)_t, \#DynalicsRelations(p)_{t-1})} \tag{7.19}$$

$$I_3^p(0) = 0 \tag{7.20}$$

This concludes the theoretical implementation of the AMAS4BigData framework to build the DREAM application. In the following, the technical implementation (software architecture) of DREAM is presented and one concrete use of DREAM is described.

[6] Dynamics relations can be active or inactive according to the state of the analytical links that carry them.

7.3 DREAM Technical Implementation: Software Architecture

This section focuses on the technical implementation (software architecture) of DREAM, setting aside the Human-Machine Interface aspect, through the description of its core architecture and the programming language used to code it.

7.3.1 DREAM's core architecture

As presented previously (cf.7.2), at its core is a multi-agent system based on the AMAS4BigData framework. During its life cycle an agent of the system executes its behaviors, triggered proactively or reactively, in order to achieve its goal or helps other agents to achieve theirs (cf.6.3).

Most of the agents behaviors, the invariant ones, use basic aptitudes. However, for the extended behaviors, which define the nature of the AMAS4BigData framework application, more complex and dedicated aptitudes are required. There are six dedicated aptitudes, called *skills*, for DREAM which are gathered in three categories:

- ***DataAcquisition* skills:** allow a percept agent to acquire (read) raw data from a data file (*FileReading* skill) or a stream of data (*StreamReading* skill) like a sensor data stream.

  This difference of the data origin types leads to different percept agent types specialized in reading one type of data, the *FilePercept* agents and the *StreamPercept* agents.

- ***DataProcessing* skills:** are pivotal in DREAM because of their use by its extended behaviors.

  The first one, the *DataNormalization* skill is attached to a percept agent so it can pre-process the raw data that it acquires (f.7.2.2).

  The second and the third skills, *CorrelationCoefficient* and *PhaseSpaceSimilarity* respectively, are the implementation of the *Dynamics Correlation* metric (cf.7.1.3) and thence used by the compute agents for the data transformation (cf.7.2.3).

- ***DataGraphManagement* skill:** is used by the percept agents to build and update the data relations graph (graph model) during the information processing step (cf. 7.2.5).

A graphical overview of DREAM's core architecture, in the form of a class diagram, is given in figure 7.18.

7.3.2 SARL: Agent-Oriented Programming Language

Many programming languages and frameworks can be used to implement a multi-agent system. Each of them has unique features suited for different kinds of applications.

To implement AMAS4BigData, I choose *"SARL"*[256] (cf. Appendix) for it is easy, fast, clear and yet powerful language, since it gathers the advantages of Java, Xtend and Scala.

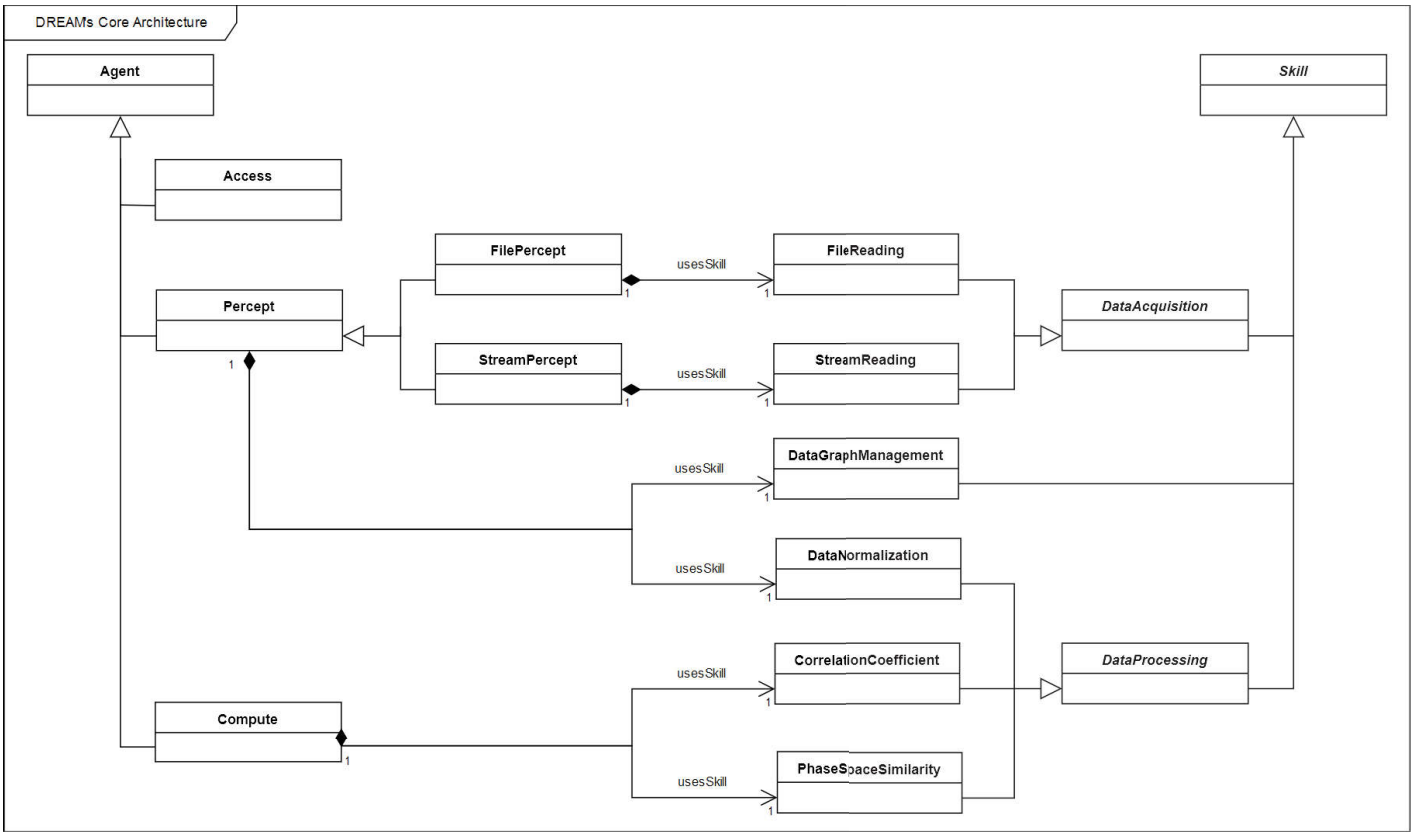

Figure 7.18 — Class Diagram of DREAM's core.

In addition, its holonic[7] view of the agents, is especially useful to run several instances of AMAS4BigData applications independently in the same container without any interference between them. In other words, all the percept agents, compute agents and their communications are contained inside the access agent of the AMAS4BigData application instance.

Definition 6. *SARL [257] is a statically-typed agent-programming language that provides the major requirements for an easy and practical implementation of modern complex software applications. dealing with concurrency, distribution, interaction, decentralization, reactivity, autonomy and dynamic reconfiguration*

The programming results in 6 200 lines of *SARL* code (11 300 Java lines after compilation) over 60 classes: 10 for agents definition, 4 of them define basic capacities and skills used by the agents, 25 for events definition, the remaining classes define various data structures and utilities required for any data processing.

Also, as an implementation of the *AMAS4BigData* framework, *DREAM* extends its code with: 300 lines of *SARL* code (700 Java lines) over 5 classes related to the *Dynamics Correlation* computation skills, plus 2 000 lines of *SARL* code (3 700 Java lines) over 10 classes dedicated to the management and display of the data graph model. Therefore, as you can notice, *SARL*

[7] An agent can contains/be composed of several sub-level agents and these sub-level agents can also be divided and so on.

considerably reduces the coding time (divided by 2 approximately), which means more time to focus on and research more about other aspects of the AMAS4BigData framework.

7.4 DREAM Uses

Besides its main purpose of discovering dynamics relation, DREAM can provide new functionalities by combining it with other tools and/or extending its internal mechanisms as described hereafter.

7.4.1 Anomaly Detection

DREAM can be used as an anomaly detection system (cf.2.2.2.2) relying on the dynamics relations discovered over time and their changes. In this context, an anomaly is defined as a non conventional change in the dynamics (behavior) of a data stream (variable). However, DREAM doesn't focus on its data stream as such, but study the relations between.

Therefore, DREAM can detect an anomaly occurring in a data stream through its relations, as a result of the disturbances this anomaly produces on the related dynamics relations punctually or over time.

Hence following the nature of these disturbances, there are two types of anomalies: the punctual anomalies (PTA) and the over time anomalies (OTA).

7.4.1.1 Punctual Anomaly

Put it simply, when the state of a dynamics relation changes too abruptly, a *(Punctual Anomaly PTA)* alert is triggered thanks to the extension of DREAM with algorithm 7.9, which is executed each time the dynamics correlation strength ($DC(A,B)$) is computed.

Algorithm 7.9: ComputeAgent.PunctualAnomalyDetection()

```
1. if DC(A,B)_t ≥ AD_THRESHOLD AND
2. |DC(A,B)_t − DC(A,B)_{t−1}| ≥ PTA_THRESHOLD then
3.     return true;
4. else
5.     return false;
```

With,

- ▷ $AD\_THRESHOLD$: is a constant, defined by the user, which represents the minimal strength that a relation should have in order to consider the detected anomalies as relevant ones. In other words, when a dynamics relation is too weak the related anomalies have a low confidence.
- ▷ $PTA\_THRESHOLD$: is another user constant that embodies the thresholds beyond which the difference between the successive strengths of the same dynamics relation is considered abnormal and a punctual anomaly is detected.

7.4.1.2 Over Time Anomaly

Unlike the previous anomalies, this kind of anomaly is detected after a period of time during which the state of a dynamics relations hugely deteriorates, leading to a momentarily loss of the situation of interest.

Consequently, the *Over Time Anomalies (OTA)* are detected thanks to an extension of the *DetectDynamicsRelation* function (cf. algo.7.5) with algorithm 7.10.

Put it simply, this extension allows DREAM to trigger an *OTA* alert when there is a strong dynamics relation that contains more than one situation of interest (SI) and the last one is a long false SI which greatly decreases the dynamics relations strength.

Algorithm 7.10: ComputeAgent.OverTimeAnomalyDetection()

1. **if** $Dynamics\_Relation\ AND\ TRUE\_SI\ AND\ COUNT\_SI > 1\ AND$
2. $\#SI \geq OTA\_LENGTH\ AND\ DC(A,B) \geq AD\_THRESHOLD\ AND$
3. $((DC(A,B) - DC(A_{SI},B_{SI})) > OTA\_THRESHOLD$ **then**
4. | **return** true;
5. **else**
6. | **return** false;

With,

- $AD\_THRESHOLD$: is the same constant defined previously (cf. 7.4.1.1).
- $OTA\_THRESHOLD$: like $PTA\_THRESHOLD$ (cf. 7.4.1.1), it is a threshold that represents the least quantity by which the dynamics relation strength should decrease because of this false SI to trigger an *OTA* alert.
- OTA_LENGTH: is the minimal length that the false SI should has in order to consider the *OTA* significant.

One use case, *"Bridge Anomaly Detection"*, of the Anomaly Detection extension of DREAM is presented in the next part (cf. 9.2.6).

7.4.2 Dynamic Data Mediation for Co-Simulation

Another use of DREAM, through its combination with the self-adaptive MAS *"AMOEBA"* [258], is the development of a new dynamic data mediation engine called *"DreAMoeba"*, fully described in [259], and briefly presented hereafter with a focus on the role of DREAM in this engine.

7.4.2.1 Co-Simulation & Data Mediation

Definition 7. *A* co-simulation *is the coupling of several simulation tools where each one handles part of a modular problem which allows each designer to interact with the complex system in order to retain its business expertise and continue to use its own digital tools. For this co-simulation to*

work, the ability to exchange data between the tools in meaningful ways is required, and the process of choosing which data go to where and translate them if needed is known as data mediation.

Technically, a co-simulation is a set of interacting simulation units, each one needs *Input* data and produces *Output* data. A simulation unit is the composition of a simulator (solver) with a dynamic system (model of a real environment characterized by a state and a notion of evolution rules). The outputs of a simulation unit might be the inputs of others.

For instance, when collaboratively solving a complex problem, like building a plane, each expert can work with his designing tool. Then, through the co-simulation, he will be able to integrate his partial solution with the others and see how its affect the overall problem.

7.4.2.2 Dynamic Data Mediation

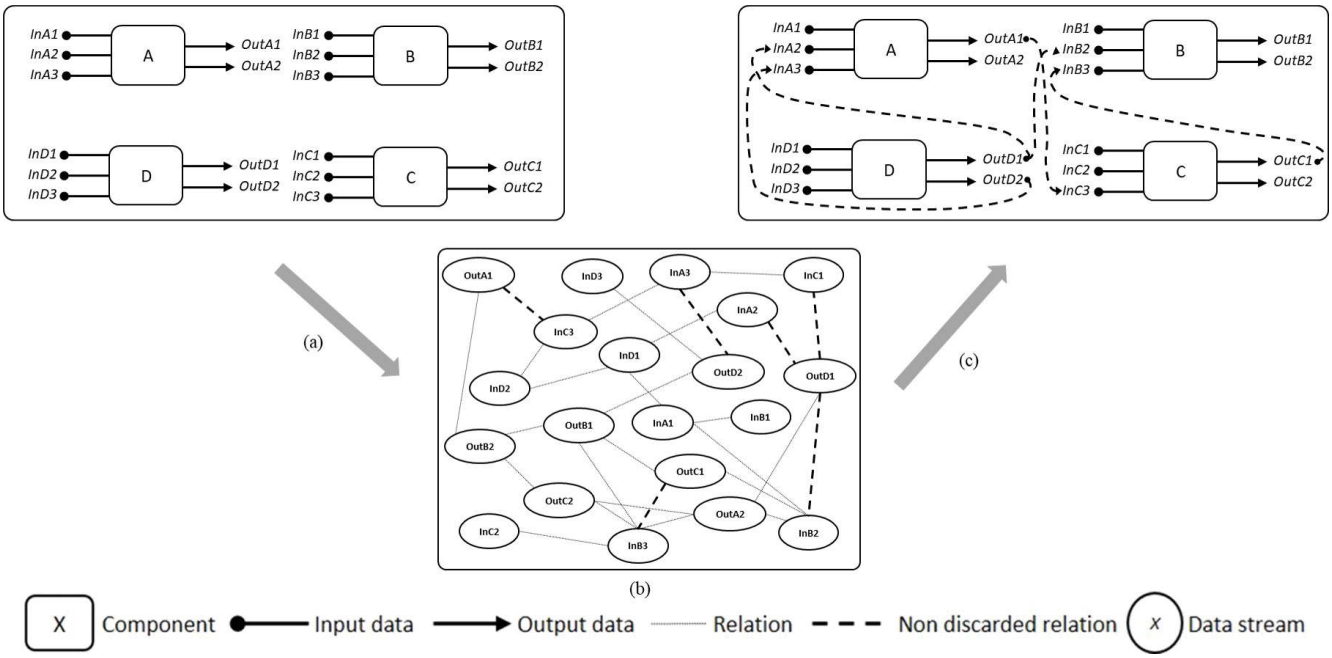

Figure 7.19 — Example of Subsystem-to-Subsystem Graph Model building with DREAM.

The matching process between the inputs and outputs of the simulation units is difficult especially in open cyberphysical environments where components join and leave the co-simulation on the fly. For this reason, we developed a new component for dynamic data mediation, called *"DreAMoeba"*, operating as follows:

1. DREAM is used as a data matcher, meaning it links the simulation units dynamically, as shown in figure 7.19, and each link is described with an *(Output, Input)* couple.
2. For each data couple query, use AMOEBA to translate the heterogeneous data form *Output* to *Input* in order to insure a semantic mediation.

DreAmoeba is applied in *neOCampus*, the ambient campus of the University of Toulouse III - Paul Sabatier (cf. 8.2.1).

7.4.3 Real-Time Eco-Correlation Detection

When a building user adopts a bad consumption behavior it reduces or invalidates the energy gain associated with the improvement of the building. Thence, behavior-based strategies to cut down energy use are among the most cost-effective on the market.

Consequently, in order to improve his energy saving, one should precisely understand the consequences of his actions so as to be motivated to change his bad energy consumption habits accordingly. As a basic example, keeping the shutters closed and turning the lights on when the sunlight can light up the room causes a pointless energy consumption.

To do so, DREAM is used to find, in real-time, better-meaning correlations between user actions and energy consumption [40], designated as *"eco-correlation"*, from huge amounts of data generated by a network of sensors and connected devices (*IoT*), and therefore present meaningful *eco-feedback*[8] (focus on relevant correlations) to help the users find where and how they can minimize their consumption.

7.5 Prospective Tools: Other Implementations of AMAS4BigData

As a conclusion to this chapter, I succinctly define some other possible tools based on the AMAS4BigData framework, showing the extent of its usability/usefulness.

7.5.1 Data Clustering

Clustering is about finding chunks of data that are similar and given the fact that the AMAS4BigData framework is good at producing data graphs, community detection algorithms seems the most suitable for clustering in this case since they aim to find groups (communities) of individuals (nodes) that are highly connected inside a community and not or poorly outside.

Thus, building a clustering tool based on AMAS4BigData can be summarized in two steps: first design the self-adaptive MAS that builds a data graph, then apply a dynamic community detection tools, like the self-adaptive MAS defined in [260] due to its high compatibility with this framework architecture.

From this definition we can distinguish two kinds of clustering: horizontal and vertical.

7.5.1.1 Horizontal Clustering

The first kind of clustering is analogous to classical clustering algorithms, such as the *K-Nearest Neighbor* algorithm, and it is relevant on data sets with low complexity (the number of data instances is far greater than the number of data features).

In this context, each percept agent represents one data instance (record, line...), and the compute agents carry out the computation of the similarity or distance measure between some of the data instances, an euclidean distance for example.

[8] Information that helps to visualize energy consumption.

Then, the exploration mechanisms (cf. 6.2.2.3) speed up the process since they focus the available computing power only on the most probable linked (similar) data couples.

7.5.1.2 Vertical Clustering

For the vertical clustering, the percept agents are associated to one data attribute (feature, variable...) and the similarity between them is computed thanks to the compute agents.

This kind of dynamics clustering is useful on time-series data, in order to find groups of data that evolve over time in the same way, like finding clusters of customers whose electricity consumption (cf. 8.2.3) is alike.

For instance, we can extend DREAM with a new type of agents, called *Cluster Agents*, which can communicate with the other agents to exploit the data graph, through the discovered dynamics relations, to find groups of connected percepts (nodes) then fuse and split on the fly consequently to the data graph changes.

7.5.2 Frequent Item-Sets Mining

The transactional data have been tackled formerly (cf. 6.2.2.2) with a short example of a rule association mining algorithm based on the AMAS4BigData framework. Nonetheless, that example was not complete because of its lack of frequent item-sets mining procedure which is provided here.

Indeed the previous example, dealt only with association rules between two items. However, association rules mining usually handles item-sets of various size. Therefore, I suggest the following incremental solution:

1. Each item is represented by a percept agent as a data stream, called here a *transactions stream*. As its name indicates, this stream provides all the transactions that its item contains.
2. The compute agents give the number of shared transactions, by selecting for each percept (item) the transactions that contains the other.
3. If $\#shared\_transactions > min\_support$, then create a new percept (item set) that combines the items or item sets, then go back to 1.
4. Else, there is no more frequent item sets.

7.6 DREAM Contributions

DREAM, as thoroughly described in this chapter, showed how one can bring the theoretical power of the AMAS4BigData framework, following its implementation steps, to a concrete application of dynamic big data analytics that can thanks to its unique features (cf. 6.5):

- be used on a wide range of domains, like the Co-Simulation (cf. 7.4.2.1);

- be deployed over various, vast, highly dynamic, and faulty data generative environments, like an IoT network;
- and find very small ore of information (weak, volatile , and flickering relations) hidden in the data, where the conventional tools are blind due to their *"whole"* way of functioning;
- present the analytics process as a white box for the user, meaning he can easily see and understand what is going on at every step of the process, and even help the system and guiding his functioning.

Moreover, DREAM draws a new way on how to see the so difficult and challenging big data analytics processes becoming within easy reach, as a result of its better understanding of the data and their evolution, thanks to its original analytical tool the *Dynamics Correlation*, and its efficient exploration of the data space.

In addition, DREAM is easily extendable, as demonstrated with the anomaly detection extension in 7.4.1, which opens the prospect of many new analytics tools based on the AMAS4BigData framework.

In conclusion, we have seen the origins of the AMAS4BigData framework, its theoretical architecture and implementation through the description of the application DREAM and it uses. The next part focuses on the experimental evaluation of the AMAS4BigData framework via several experiments of DREAM.

AMAS4BigData
Adaptive Multi-Agent Systems
for Dynamic Big Data Analytics

Experimental Evaluation

Abstract

NOW that theoretical and technical foundations of the AMAS4BigData framework are set, this part provides an experimental evaluation of the framework through its implemented tool, *DREAM*, by means of several distinct experiments conducted on various data sets. The experiments are evaluated with specific criteria and the results are analyzed in order to explain the framework behavior and assess its efficiency.

In this scope, chapter 8 presents the experimental protocol of the framework evaluation, through the description of all the data sets used to conduct the experiments on the framework and the definition of all the criteria employed to appraise the framework capabilities.

Then, chapter 9 presents the various experiments and their settings: *DREAM*'s parameters, the data sets used and the chosen evaluation criteria. Finally, it presents and discuss the results of the previously defined experiments, and gathers them to construct an overview of the framework performances, points out its strength and weaknesses, and consequently discusses when and how to use the framework for the sake of an optimal dynamic big data analytics process.

8 Dynamic Data Sets & Experimental Protocol

Before studying the actual experimental evaluation of the AMAS4BigData framework, presented in the next chapter, we must understand the components used to establish the experimental protocol: the dynamic datasets and the evaluation criteria.

The data are the nerve center of the experimental evaluation of the AMAS4BigData framework. This chapter presents and describes all the datasets used to achieve this goal. Each datasets has its own features that aim to fulfill different evaluation purposes.

Considering that AMAS4BigData is evaluated by means of the application *DREAM* (cf. 7), all the data are time-series[1] and come in two categories: the synthetic datasets and the real world datasets.

[1] A time-serie is a sequence of data points, over time, related to one entity or one entity feature.

8.1 Synthetic Data Sets

The first category of datasets embodies artificially generated data that look like real *IoT* data. Thence, these synthetic datasets are collections of data streams (time-series) that hold useful characteristics to evaluate the framework, as explained in this section.

8.1.1 Description & Motivation

One may legitimately ask *"why use synthetic data, which may be skewed, if there is various real datasets?"*. Despite the ever growing available real data, to fully evaluate this new analytics framework with taking into account its original capabilities, the datasets to be used should exhibit additional features to appraise such capabilities.

Unfortunately, even if some of these required features may be available in real datasets, they are not all easily or freely available at the same time. So, I define them here and I describe the process that makes them after that.

- **Complexity**: since DREAM aims to find complex data relations (non-linear dependencies), the data it analyzes in the scope of its evaluation should contain such complex relations. These complex relations are expressed by non-linear shapes in the data scatter plots (cf. 7.1.1).

  The synthetic datasets are made to enclose at the same time non related data, straightforwardly related data (linear dependencies), and intricately related data as shown in figure 8.1, whereas regular real world data often does not.

- **Dynamicity**: the second feature of the synthetic datasets, serves to show the ability of the framework to handle changes occurring in the behavior (dynamics) of the data it processes.

  These changes are translated as substitutions of the default data dynamics, during some intervals, with new or random dynamics. Consequently, it results in erratic data relations, meaning that two data streams maybe be correlated during a given period of time and maybe not at an other time.

- **Benchmarking**: a huge drawback of the real world datasets is their lack of labeling or annotation. In other words, we have data to work with but we do not know what results we should expect.

  Therefore in addition to the base data, the synthetic datasets must furnish a list of all existing dynamics relations, which can be compiled in forms of a data relations graph (cf. fig 8.3), thanks to the process that produces them described hereinafter.

8.1.2 Data Factory

In order to automate the generation of synthetic datasets and manage their features, I designed a process called *"Data Factory"*. The data factory is a Python script which builds KNIME workflows (cf. 3.3.1) that generate the synthetic datasets. These workflows are

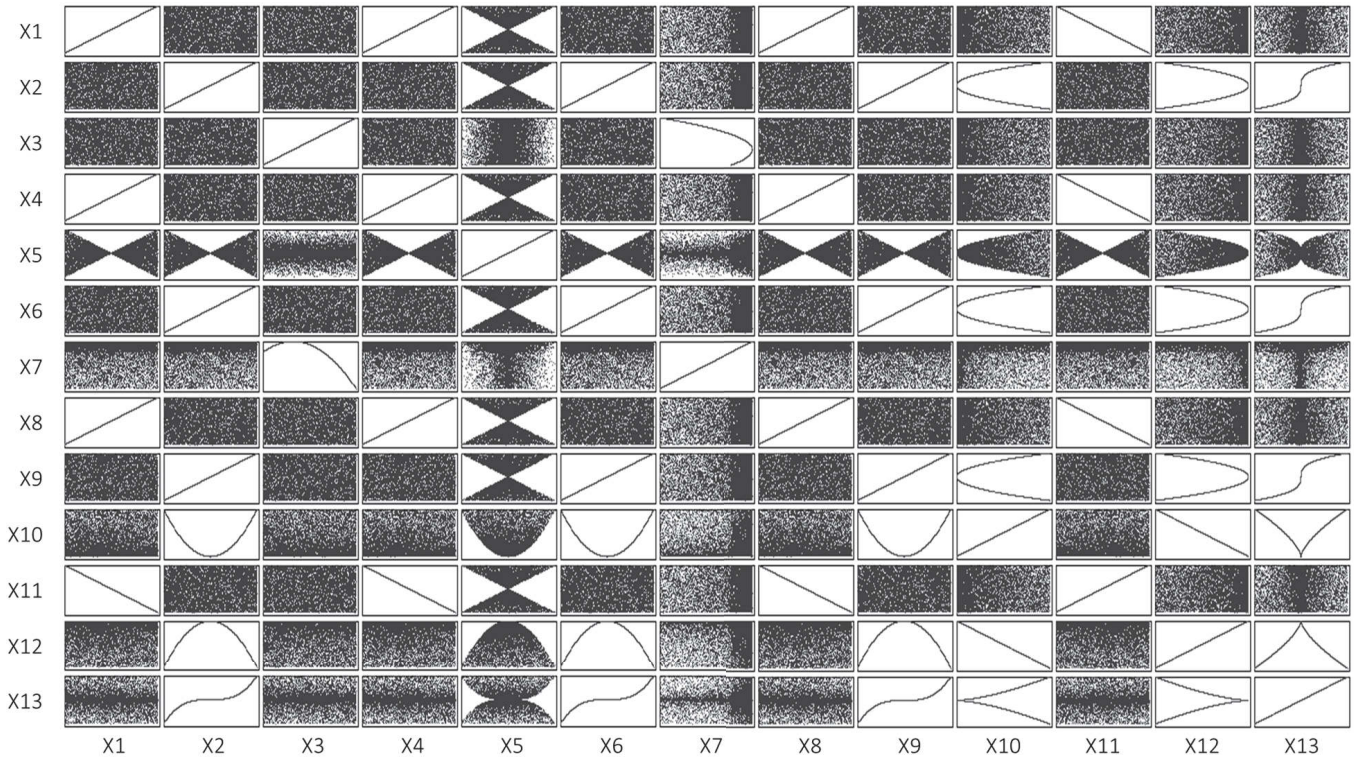

Figure 8.1 — Scatter matrix of a synthetic dataset.

designated as *"datasets generators"* and produces different synthetic datasets with the same features (cf. 8.1.1).

8.1.2.1 Data Factory Process

The data factory assembles datasets generators in shape of poly-tree KNIME workflows, as illustrated in figure 8.2. Therefore, a datasets generator is a set of task nodes, that produce a data stream, distributed over several layers and nodes of one layer are connected randomly to nodes of the next layer.

Once the data factory builds a datasets generator (cf. fig 8.2), it constructs the corresponding data relations graph (cf. fig 8.3), which is used for benchmarking purposes (cf. 8.1.1), following one simple rule: two nodes A and B have a dynamics relation if A is an ancestor of B or A and B have one common ancestor C.

Moreover, the data factory is a configurable process where the user can choose the number of roots, the number of layers, the number of nodes per layer, the number of data points (instances) that the datasets generator must produce, and the functions (transformation and combination) used to produce new data streams (cf. 8.1.2.2).

8.1.2.2 Data Sets Generator

After its manufacturing, a datasets generator (cf. fig 8.2) can start producing the data streams from the first layer (the roots) to the last one (the leaves) following the connections

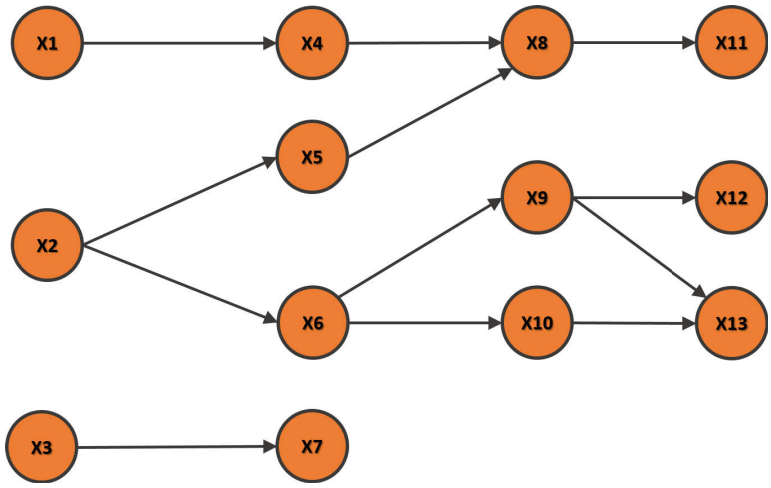

Figure 8.2 — Poly-tree data generator workflow.

between them.

A connection between two nodes indicates that the data produced by the parent node take part in the data production task of the child node, as described below. In addition, the task nodes are of two kinds:

- **Root nodes**: are special nodes present in the first layer. They produce random data streams with different distributions, in order to avoid any correlation between them, and by extension to prevent the occurrence of any untraceable relation between their children nodes data streams.

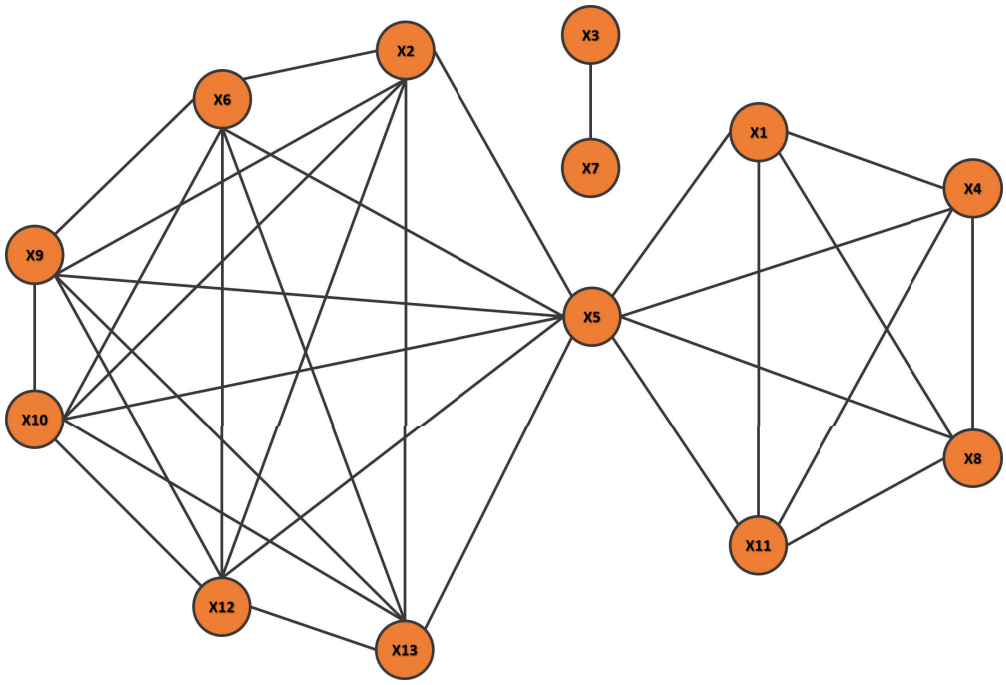

Figure 8.3 — Generated data relations graph.

For instance, if the roots data streams were based on sinusoidal functions, their cyclic behavior may cause unwanted correlations following their wavelength.

- **Stream nodes**: are regular nodes present in the next layers and combine their parents data streams linearly or, more importantly, non-linearly and can introduce some noise and randomness.

When a stream node performs its task, it transforms the input streams thanks to a transformation function chosen randomly, be it linear or not, and combine them with another randomly chosen function. Thus, one generator can produce several datasets that relate the same data in different ways (different dynamics) while preserving the same features.

8.2 Real-world Data Sets

The second category of datasets, the real-world datasets, are used to evaluate and discuss the realism and several real-world specific proprieties of the AMAS4BigData framework in the next chapters.

8.2.1 neOCampus Data

For a real time evaluation of the framework on a real sized smart city infrastructure, I rely on the IoT data of the *neOCampus* project, the ambient campus of the University of Toulouse III.

8.2.1.1 neOCampus Project

The *neOCampus* project [261], [262] is supported by the University of Toulouse III. Its aim is to demonstrate the skills of researchers of different domains of the University towards the design of the campus of the future. Three majors goals are identified: ease the life of campus users, reduce the ecological footprint, control the energy consumption.

The campus is seen as a smart city, a complex system, where eventually several thousands of data streams come from heterogeneous indoor and outdoor sensors (CO2, wind, humidity, luminosity, human presence, energy and fluids consumption...). Artificial intelligence techniques are used to understand the aims and behaviour of citizens from manually selected subsets of data.

8.2.1.2 neOCampus IoT Data

Student activity is one of the main generator of neOCampus IoT data. This IoT is composed of a wide range of pervasive ambient sensors and effectors that continuously produce heterogeneous data across the campus (classrooms, campusFab, library...). An example of one week data recorded from 17 sensors is shown in figure 8.4.

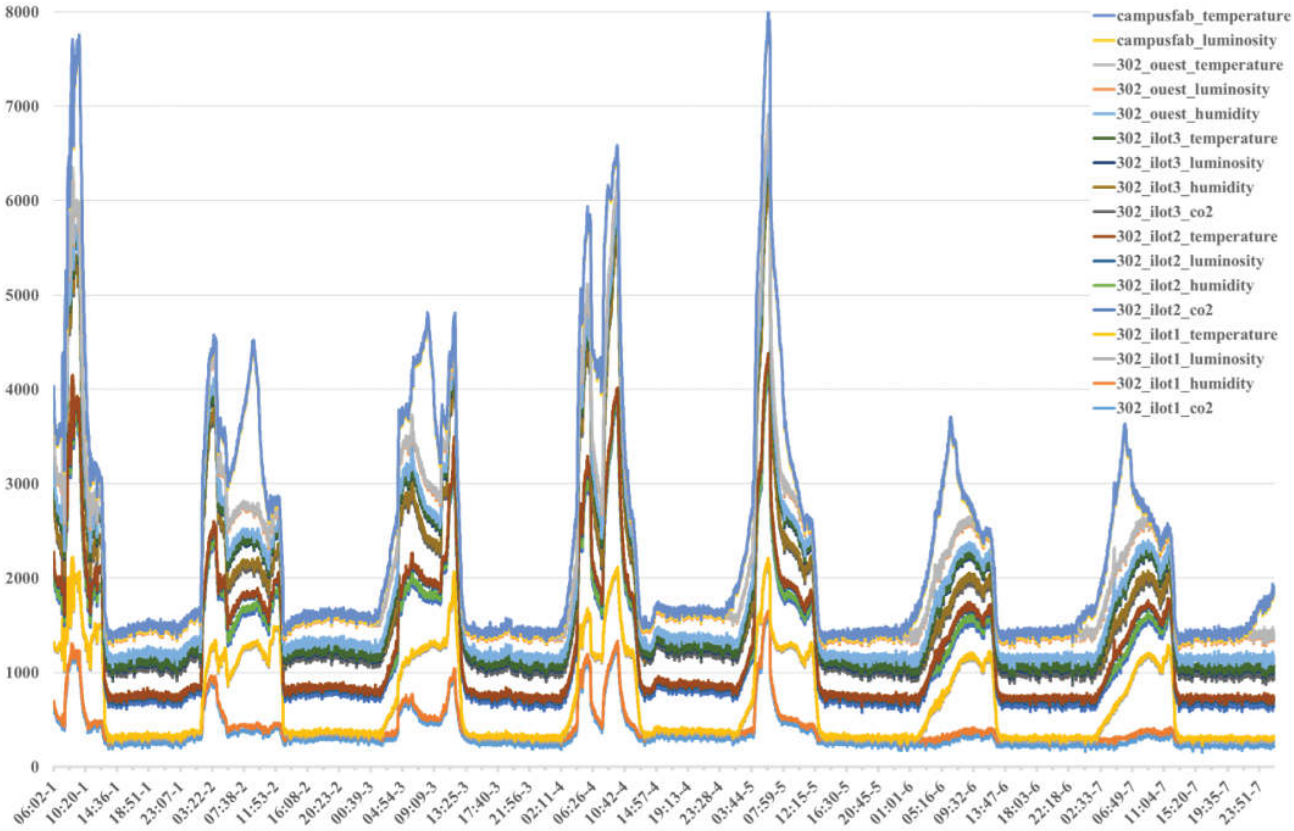

Figure 8.4 — One week of neOCampus IoT data.

The neOCampus IoT data used here, constitutes two kinds of dataset: a regular small offline dataset, the produced data introduced previously (cf. 8.4), and an online real-time dataset, available for researchers via neOCampus network.

Moreover, the amount of the produced data is perpetually increasing, since neOCampus IoT is being iteratively extended with new devices[2].

8.2.2 Bridge Data Set

The bridge dataset is a collection of twelve data streams from four accelerometers located under the four ends of a two-lanes bridge in Japan. Each accelerometer recorded the vibrations of the bridge, every 5 milliseconds, over the three dimensional space axes (X, Y, Z) during a 10 minutes window at night time, as shown in figure 8.5.

This dataset was used in the course of a 3-month research internship I did at the *National Institute of Informatics (NII)-Tokyo*. Therefore, due to its ownership by the Japanese government all information about the bridge itself, like its location and dimensions, are confidential.

That internship aimed at designing a real-time anomaly detection tool, which could be used to trigger an alert when abnormal vibrations that may hint at a structural weakness are

[2] At the time of the writing, there are 86 data sources available online.

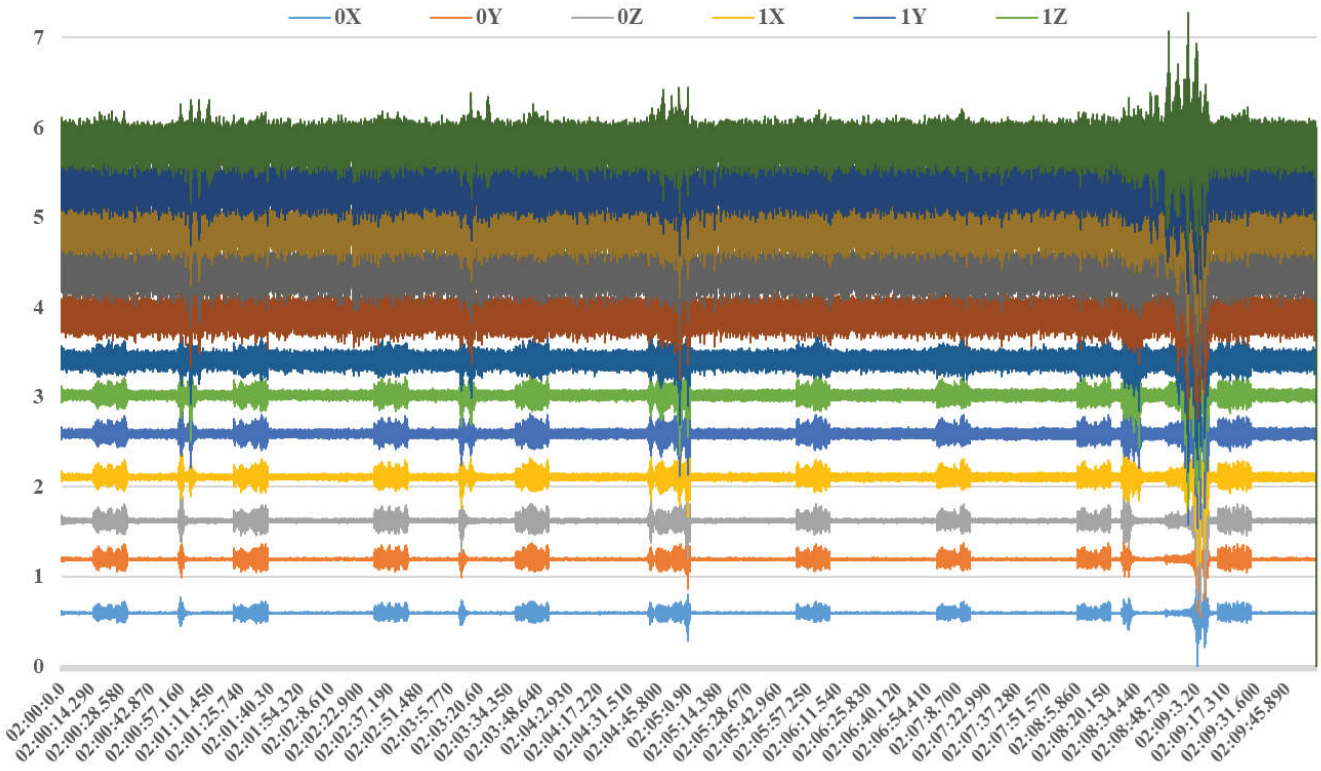

Figure 8.5 — Normalized bridge data.

detected. Further details are given in the *Bridge Anomaly Detection* experiment (cf. 9.2.6).

8.2.3 UCI Data Sets

The UC Irvine Machine Learning Repository (UCI) [263] provides relevant datasets with high number of variables, that highlight the advantage of using the AMAS4BigData framework to analyze huge datasets. From the UCI repository, I took the two following time-series datasets:

- **Ozone Level Detection** [264]: is a mid-sized datasets containing 72 variables and 2536 lines (days), with some missing values replaced by the mean, from 1998 to 2004 at Houston, Galveston and Brazoria. This dataset records atmospheric data (temperature, humidity, wind direction and speed, pressure, solar radiation, ozone concentration...).

- **ElectricityLoadDiagrams20112014** [265]: is a large dataset, that records the electricity consumption in kW of 370 customers from 2011 to 2014 every 15 minutes, which leads to 140256 lines. This particular dataset is used for testing the framework scaling capability, due to its big size and the high number of data relations it contains because of the numerous consumption dynamics similarities between the customers.

8.3 Evaluation Criteria

An evaluation criterion is a metric that measures the performance of the framework under one point of view at each step of its analytics process (analytics cycle). In addition, even though all the evaluation criteria used to design and test the framework are defined hereafter, only some of them are actually part of an experiment setting and discussed in the next chapter.

8.3.1 Agents Evaluation

The first group of evaluation criteria concerns the agents of the framework, more specifically, the criticality of the percept agents and usage of compute agents.

- **Percepts Criticality**: the criticality of a percept agent expresses its need for a new compute agent (computing resource) in order to analyze an analytic link.

  This criterion is computed with the criticality function (cf. 6.3.4) and given for each percept at each step by the triplet $(Min\_Criticality, Mean\_Criticality, Max\_Criticality)$ corresponding respectively to the minimal, the mean, and the maximal criticality value among all the percept agents.

  Thus, the percepts criticality criterion aims to study, from the percept agents point of view, the fairness of the compute agents allocation and detect any resource starvation in the worst case scenario.

- **Resources Usage**: is a couple value $(CAC^3, PACA^4)$ of the compute agents count, the number of computing resources used by the framework, and the percentage of allocated compute agents from that pool of resources.

  Since the AMAS4BigData framework is designed to be elastic (cf. 4.4), this criterion exhibits its ability to acquire new computing resources, when the overall available computing power is sufficient, and release them when they are not needed anymore.

8.3.2 Exploration Evaluation

The following evaluation criteria show how the AMAS4BigData framework explores the data relations space and how efficiently it manages its available computing power.

- **Active Relations**: when the CAC^3 and the $PACA^4$ of resource usage criterion, defined previously, are combined they give at each analytic cycle the number of data relations (analytic links) currently analyzed, and by extension their percentage over all the possible relation ($n(n-1)/2$, where n is the number of input data streams).

- **Relations Coverage**: indicates the percentage of explored relations from the beginning of the analytics process until the current step.

[3] Compute Agents Count.
[4] Percentage of Allocated Compute Agents.

Moreover, when the criterion reaches 100%, meaning all the possible relations have been visited at least once (full coverage), it is reset to the active relations percentage so the user can observe how quickly and how many time the framework re-explore all the possible relations.

- **Punctual Usefulness**: when a compute agent is allocated to analyze a analytic link (data relation), it may or may not change the state of the link. When it does it is considered *useful* because it brought novelty in the exploration of the data relations space. Thus, this criterion computes the percentage of such useful compute agents for each analytic cycle.
- **Overall Usefulness**: gives the percentage of the compute agents that have been *useful* at least once since the beginning of the analytic process and it is reset to 0% when it reaches 100%, meaning that all the compute agents have been useful once.

8.3.3 Information Retrieval Evaluation

The criteria of this group are dedicated to the *DREAM* application, since they measure the relevance of the dynamics relations discovered. Thence, these criteria are those of the *Information Retrieval* field [266].

Given,

- *True Positive (TP)*: the number of true dynamics relations from the relations found,
- *False Positive (FP)*: the number of relations found (considered true) but that are not,
- *True Negative (TN)*: the number of not found relations and are truly not dynamics relations,
- *False Negative (FN)*: the number of not found relations which are actually dynamics relation.

The information retrieval evaluation criteria are defined as follows:

- **Precision**: estimates from all the relations found by *DREAM* how many are true dynamics relations (cf. fig.8.6).

$$Precision = \frac{TP}{TP + FP} \tag{8.1}$$

- **Recall**: indicates the ability of *DREAM* of finding all the dynamics relations among all the data relations space (cf. fig.8.6).

$$Recall = \frac{TP}{TP + FN} \tag{8.2}$$

- **F1-Score**: is a combination of the former two measures and gives an overall measure of the dynamics relations discovery relevance. The evolution of *F1-Score* given *Precision* and *Recall* is illustrated in figure 8.7.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8.3}$$

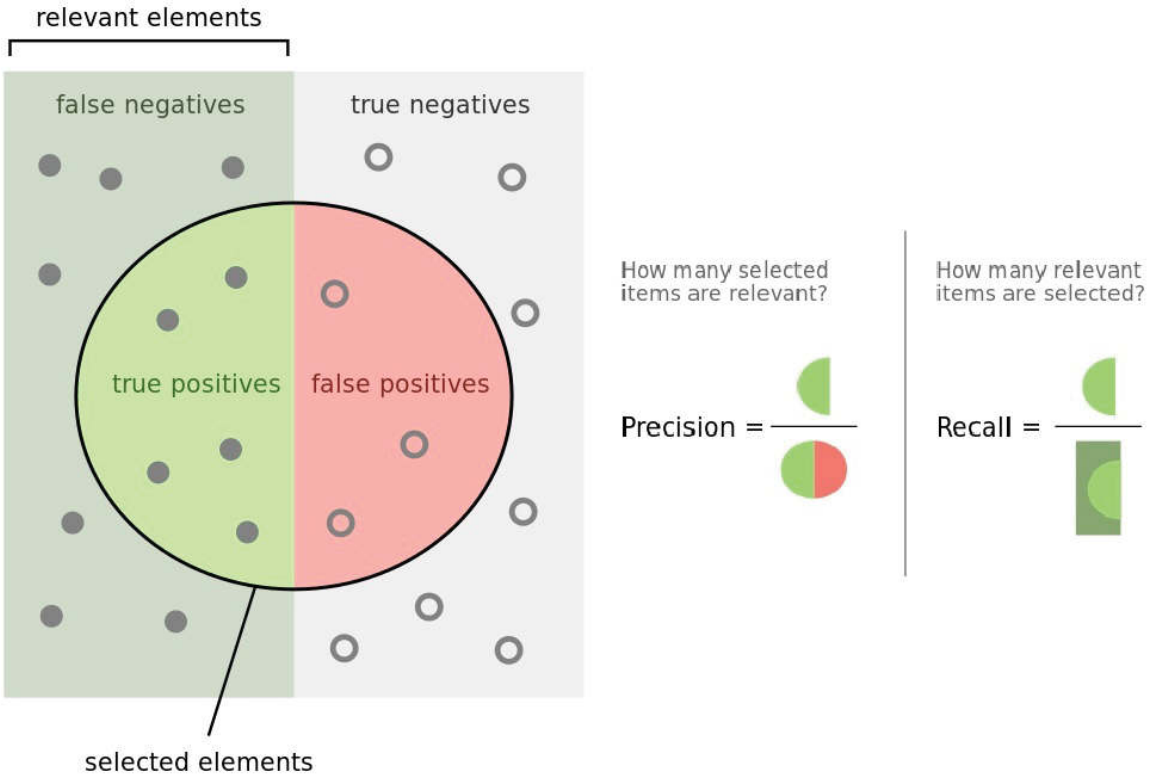

Figure 8.6 — Precision & Recall [267].

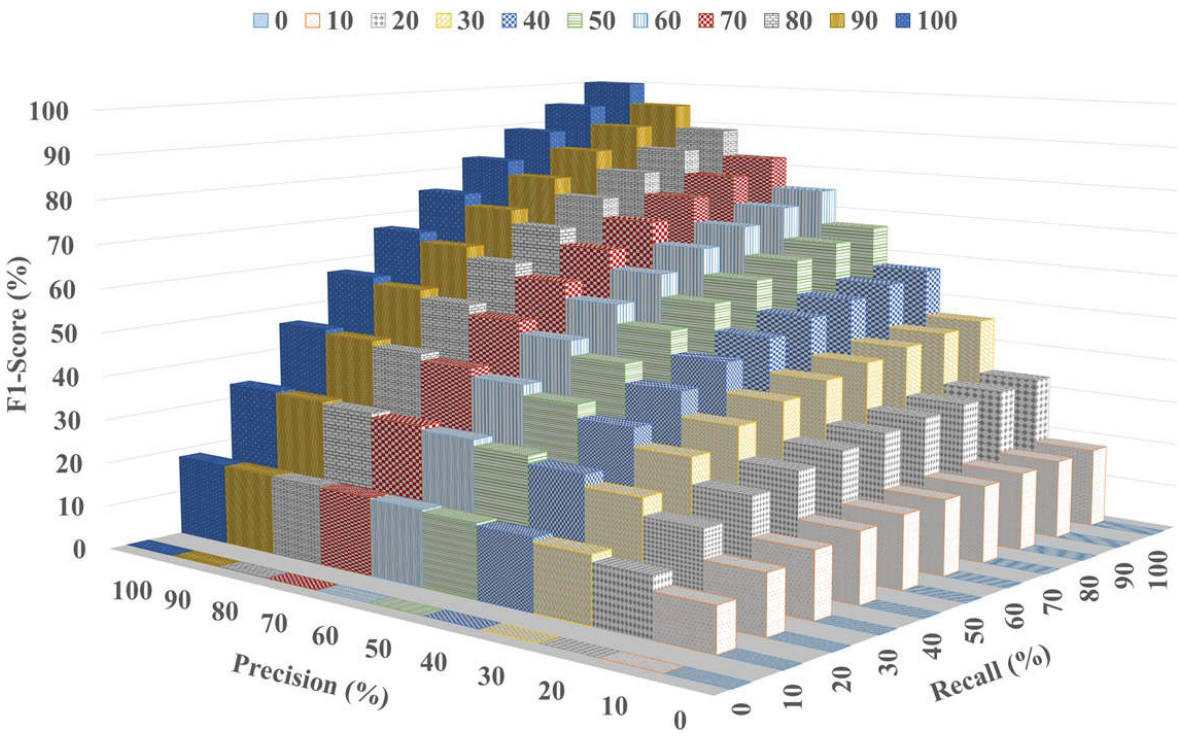

Figure 8.7 — *F1-Score* according to *Precision* and *Recall*.

8.3.4 Space & Time

The last evaluation criteria are inherent to any algorithm evaluation, since they assesses the processing time, speed, and memory usage of the framework.

- **Elapsed Time**: as designated, it is the time spent by the framework to analyze the input data. In order words, the elapsed time of the analytics process.
- **Processing Speed**: is the number of analytics cycle (data lines) over the elapsed time. It measures how fast the framework processed the data and helps to study the progression of this speed.
- **Memory Usage**: gives the overall memory used by the framework at each step of the analytics process.

In conclusion, the AMAS4BigData framework is evaluated on several very different datasets with various criteria. Each dataset, with its unique features, and each evaluation criterion enables us to study the performances of the framework and assess its different capabilities, as described in the next chapters.

9 Experiments & Evaluation

Seeing that AMAS4BigData is a theoretical/abstract framework, its evaluation is transposed to its main implementation, the application *DREAM* (cf. chapter 7). Therefore, all the experiments presented henceforth are carried out by *DREAM*.

Every experiment has a setting and is undertaken several times in order to produce statistically significant results, presented in the last section. A setting is the selection of one data set, from the data sets presented in the previous chapter, to be processed by *DREAM*, the definition or redefinition of *DREAM* parameters and a set of evaluation criteria defined in the experimental protocol chapter (cf. 8.3).

9.1 Experiments Setting

Now that all the prerequisites are set, we can go through the configuration of all the experiments used to evaluate the AMAS4BigData framework.

Technically, an experiment is one execution of the *DREAM* application, with the default or overridden parameters (cf. 6.4), on a given data set and outputs the values of the chosen evaluation criteria (cf. 8.3). The experiments are organized based on their data sets as presented next.

9.1.1 Default Setting

- *Exploration Mechanisms Parameters* (cf. 6.4): $P_{TR} = 1$, $P_{SL} = 1$, $P_{RL} = 1$, $X\_factor = 30\%$, $MaxIdleDelay$ (cf. 6.5) $= 10 + RandomVal(10)$;
- *Links Life Cycle Parameters* (cf. 22): $min\_eval = 0$, $LinksRemovalTime = 50$, $LinksDeactivationTime = 30$;
- *Data Transformation Parameter* (cf. 9): $min\_len = 20$;
- *Resources Management*: automatic (cf. 6.3.2.5);
- *Hardware Configuration*: all the experiments are carried out on a machine with an Intel Core i7-4790 CPU @ 3.60GHz and 32Gb DDR3 RAM.

  It should be noted that at the time of the writing, some experiments were conducted using two HPC[1] platforms dedicated to very large data sets processing: *OSIRIM*[2] [268] and *CALMIP*[3] [269]. However due to their small number and lack of repetition (consistency), these experiments do not constitute a sufficient working base to make any conclusion from their results regarding the framework evaluation.

9.1.2 Synthetic Dynamics Experiments

This category of experiments, the *"synthetic experiments"* for short, are made to evaluate the output relevance of the AMAS4BigData framework with three slightly different settings presented below. Hence, the experiments are organized in three groups corresponding to each setting.

Nevertheless, the synthetic experiments are repeated several times over the same range of synthetic data sets 8.1.1. This range is composed of 10 synthetic data sets that mimic one week of real world IoT data generated every 30 seconds, leading to 20160 instances. Additionally, the number of data streams increases from 20 to 200 alongside to 15%-19% of dynamics relations from all the possible links.

[1] High Performance Computing (cf. 3.2.2.4).

[2] *Observatory of Systems Information Retrieval and Indexing of Multimedia contents* is a platform the *Institut de Recherche en Informatique de Toulouse* and consists of a 928 core computing cluster.

[3] A big data computing center, located in Toulouse, that provides a platform with 13 464 cores.

Therefore, the number of links to analyze (exploration space) and the amount of relevant relations of the data sets are exponentially increasing, as illustrated in figure 9.1, which allows us to study the framework scalability.

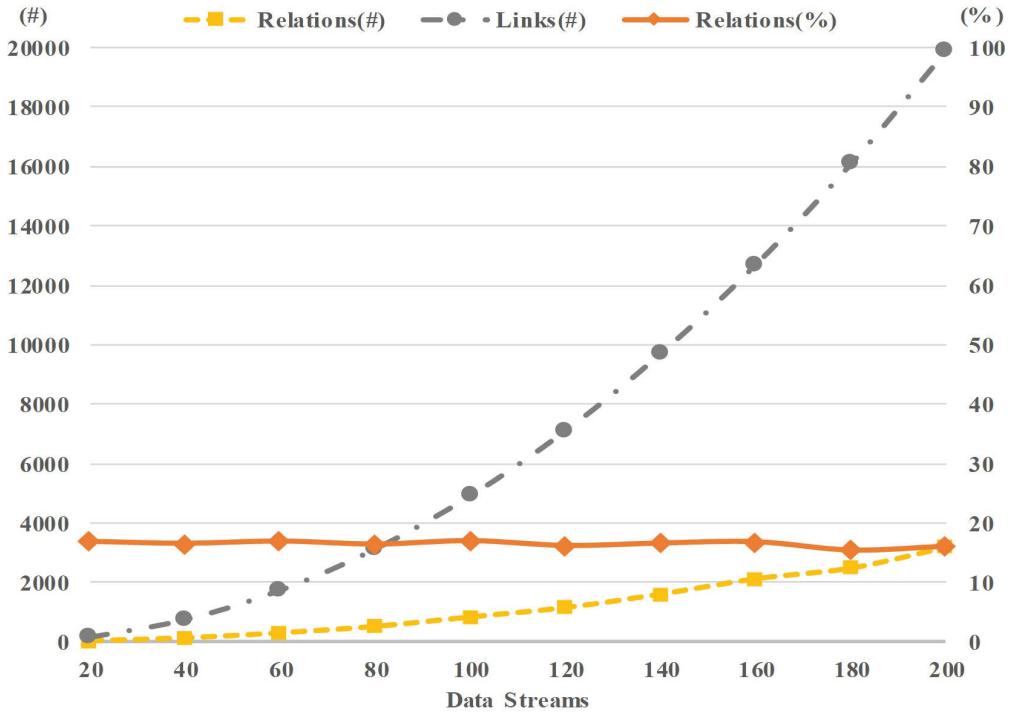

Figure 9.1 — Synthetic data sets size.

- **Baseline Experiments**: use the default configuration and serve as a baseline to compare the results of the experiments coming next;

- **Random Activation Experiments**: override the default setting by resetting all the exploration mechanisms probabilities to $P_{TR} = 10^{-1}$, $P_{SL} = 10^{-1}$, $P_{RL} = 10^{-5}$. Thence, the exploration mechanisms are triggered considerably less than in the base line experiments;

- **Resources Degradation Experiments**: aim to investigate the impact of decreasing the amount of compute agents on the output relevance. Thus the corresponding setting is kept at the default setting, but with a fixed resources management, whose number of computing resources decreases while the size of data sets increases.

- **Data Perturbation Experiments**: are the final synthetic experiments and are designed to evaluate the ability of the AMAS4BigData framework to withstand perturbations, using noisy data sets.

  These data sets are the same ones used in the baseline experiments with the insertion of three small ranges (5% of the data instances of the base data sets) of random data, called *"perturbations"*, at the first, the second, and the third quarter of the data time line (cf. figure 9.7).

Except for the perturbation of the data sets, these experiments use the same setting as the baseline experiments setting.

Finally, for the evaluation criteria (cf. 8.3), according the study purpose of the synthetic experiments, we select the following: the *Information Retrieval* criteria (*Precision*, *Recall*, *F1-Score*), the *Resources Usage* criterion and *Space & Time* criteria (*Elapsed Time*, *Memory Usage*).

9.1.3 IoT Experiments

The former experiments focus on the evaluation of the analytical process at its end mainly, through its final output. However due to their real-world data, the three IoT experiments are designed to examine the analytical process at each step from the beginning, so we can study the evolution of the AMAS4BigData framework needs and performances.

These three experiments have different settings, each one suited to its corresponding data set size (small, medium, big) as described hereafter, for the sake of a reasonable trade off between the resources consumption and the framework performance.

- **neOCampus Experiment**: uses a small data set, the off-line *neOCampus* data set (cf. 8.2.1), with the default setting;
- **Ozone Experiment**: applies the framework on a medium sized data set, the *Ozone Level Detection* data set (cf. 8.2.3). The setting applied is the default one with $X\_factor = 10\%$, so the experiment requires less computing resources;
- **Electricity Experiment**: pushes the framework capabilities the farthest thanks to the biggest data size available, the *ElectricityLoadDiagrams20112014* data set (cf. 8.2.3). This experiment uses an even lesser amount of computing resources, by decreasing the X_factor to 1% and the exploration mechanisms probabilities to the same ones as the random activation experiments.

The IoT experiments evaluate the AMAS4BigData framework through the following criteria: all the *Exploration* criteria, the *Resources Usage* criterion and *Space & Time* criteria (*Elapsed Time*, *Memory Usage*).

9.1.4 Bridge Anomaly Detection Experiment

The final experiment is unique considering that it attempts to compare the framework output, anomalies in this case (cf. 7.4.1), to the output of a conventional anomaly detection technique, called here *Fourier Outlier Detection (FOD)*.

More specifically this is done thanks to an Outlier Detection, with a robust covariance estimation (Minimum Covariance Determinant) using Mahalanobis distances relevance, applied on the Fast Fourier Transform (FFT) of the data streams, and then the transformation of the outliers back to the data streams space in order to pinpoint the corresponding anomalies.

Definition 8. Fast Fourier Transform (FFT) *is an optimized Discrete Fourier Transform (DFT) [270]. This transform breaks a signal down to the frequencies, with their amplitude, of the sinusoidal oscillations that compose it over the measured time period. For instance in figure 9.2, the signal contains 3 distinct dominant frequencies.*

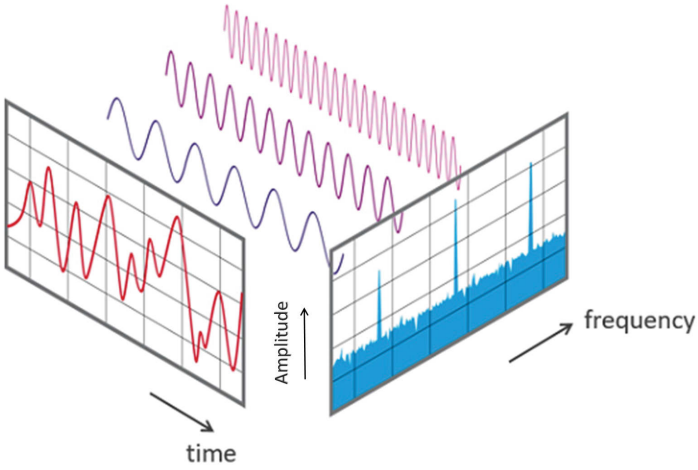

Figure 9.2 — View of a signal in the time and frequency domain (Fourier transform) [271].

Definition 9. The Minimum Covariance Determinant estimator (MCD) *is a robust estimator of covariance, that exploits the Mahalanobis distances to derive a measure of outlyingness (cf. figure 9.2). It can be used to estimate the covariance matrix of highly contaminated data sets, and thence detecting outliers more efficiently even in data sets containing a lot of outliers [272].*

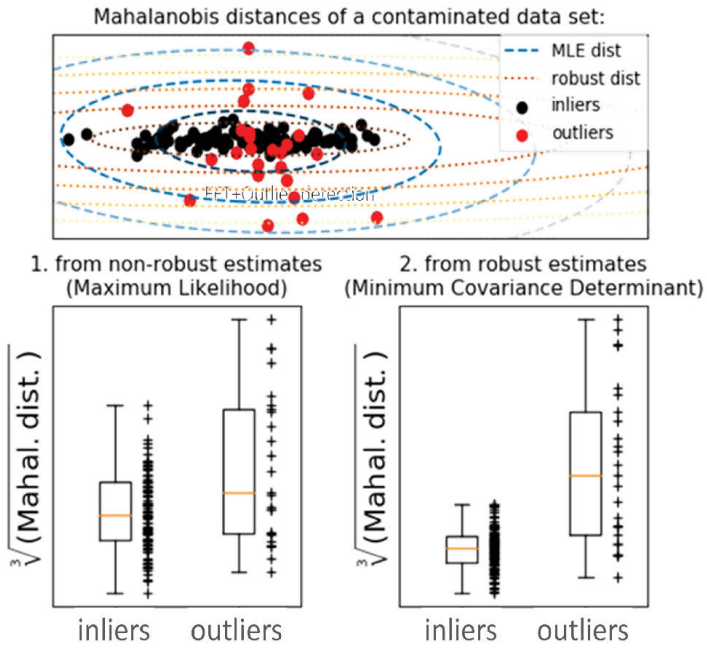

Figure 9.3 — Outlier Detection with robust covariance estimation using Mahalanobis distances relevance [273].

9.2 Experiments Results Discussion

The results of the previously defined experiments are gathered in different categories to discuss one aspect of the framework at the time.

9.2.1 Output Relevance

The first category of experiments results, the *output relevance*, studies the relevance of the discovered dynamics relations by means of the synthetic experiments starting with the baseline experiments, then resources degradation experiments, the random activation experiments after that, and finally the data perturbation experiments. The numeric results of these experiments are given in tables after their discussion.

9.2.1.1 Baseline

To begin, the results of the baseline experiments are shown in figure 9.4. As we can see, the relevance of the framework output expressed with the *precision*, *recall*, and *f1-score* is quite high with a slight decrease broadly speaking.

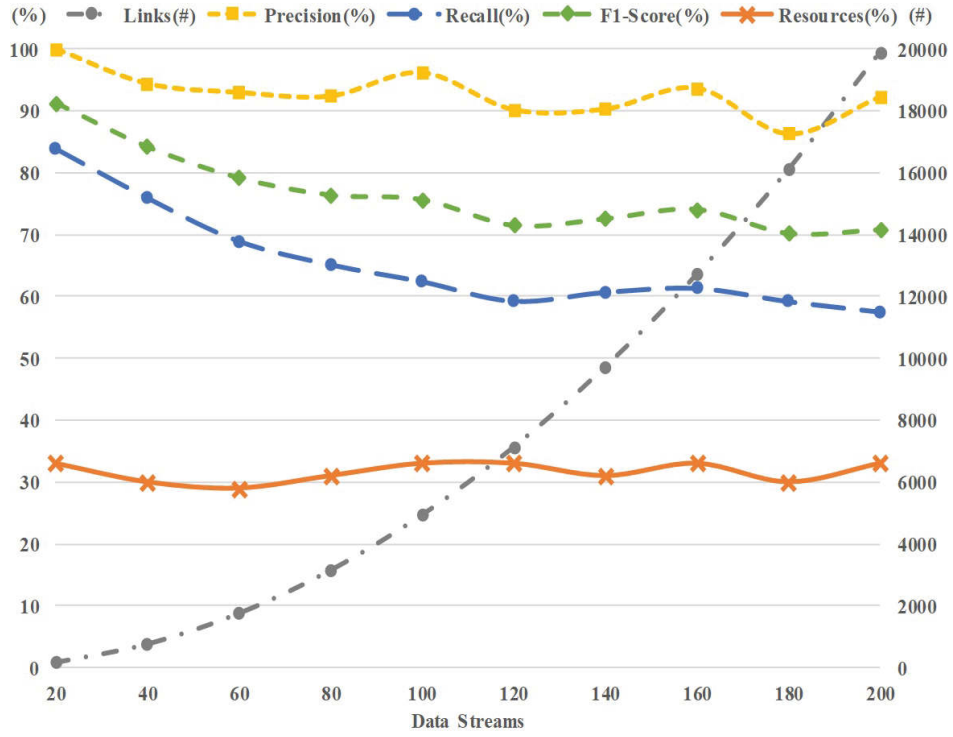

Figure 9.4 — Baseline experiments: Relevance of the discovered relations.

In concrete terms, the *precision* is at the top (100%) for the smallest synthetic data set (20 data streams) and slowly decreases, also designated as graceful degradation, while experimenting on exponentially bigger data sets (cf. figure 9.1), with an average reduction of 0.86% and an average standard deviation of 1.09% leading to 92.27% *precision* for the last and biggest synthetic data set.

As a reminder, *precision* indicates the correctness of the discovered relations, thanks to the benchmarking feature of the synthetic data sets (cf. 8.1.1). Thus given the resulted *precision* (cf. figure 9.4), the framework produces accurate output. Moreover, the slow loss of *precision* of the framework can be explained by two intrinsic traits of the synthetic data sets:

- On one hand, the exponential growth of the data sets increases the occurrence probability of not benchmarked dynamics relations. In other words, some of the discovered relations by the framework may not appear in the benchmark even if they are true and consequently loosing some *precision*. This would of course be a scientific observation issue, and not a problem with the framework.

- On the other hand and most likely, the generated dynamics relations are volatile[4], meaning dynamics relations can disappear and reappear over time. Then, by the time that the framework finds a dynamics relation, it might temporally disappear misleading the framework evaluation.

Besides, this explains the regain of *precision* for the data sets of 100, 160 and 200 streams. In these data sets, the dynamics relations are less volatile and better benchmarked, hence the gain of *precision*.

Furthermore, as illustrated in figure 9.4, the measurement of finding all the dynamics relations, the *recall*, points out that the framework discovers most of the dynamics relation inside small data sets, then starts loosing some relations at a rate of 2.94% to reach 57.49% *recall* with an average standard deviation of 0.61%.

The decline of the *recall* is a direct consequence of the *precision* loss and its important ripple effects on the data space[5] exploration:

- In the first place, there is a small shared pool of computing resources, so the ones used to find supposedly false dynamics relations leave fewer resources to find the relations still remaining.

- In the second place, the framework agents cooperate to steer the analysis toward the links close to relevant ones mostly, since they are more likely to be relevant as well. So, the supposedly false relations misguide the agents in choosing the right links to analyze, leading to a waste of resources in exploring less relevant areas in the data space, hence a loss in *recall*.

Finally, due to its compound nature, the global relevance criteria *f1-score* follows the same pattern as for *precision* and *recall*. Nonetheless *f1-score* is closer and more similar to *recall* compared to *precision*.

[4] Not systematically present during all the time line of the data set.
[5] All the possible links between the data streams.

Data Streams	Resources(%)	Precision(%)	Recall(%)	F1-Score(%)	$\sigma_{Precision}$(%)	σ_{Recall}(%)	$\sigma_{F1-Score}$(%)
20	33	100	83.96	91.28	0	2.95	1.92
40	30	94.43	75.98	84.21	1.94	0.37	0.86
60	29	93.01	68.96	79.2	1.12	1.17	0.94
80	31	92.41	65.15	76.42	1.19	0.67	0.55
100	33	96.16	62.41	75.69	0.92	0.12	0.03
120	33	90.24	59.24	71.53	1.99	0.3	0.03
140	31	90.33	60.67	72.59	1.19	0.19	0.02
160	33	93.69	61.31	74.12	0.86	0.02	0.11
180	30	86.38	59.19	70.25	1	0.1	0.07
200	33	92.27	57.49	70.84	0.7	0.2	0.05
				Average	1.09	0.61	0.46

Table 9.1 — The numeric results of the baseline experiments.

9.2.1.2 Resources Degradation

Now we move to the degradation experiments, which focus on showing how reducing the amount of computing resources have an effect on the output relevance.

The experiments results, given in table 9.2, exhibit the same evolution as the baseline for the smallest data sets (20 and 40), because they use the same quantity of resources. Then, the percentage of compute agents starts dropping at an average rate of 2.58%, and as illustrated in figure 9.5, the *f1-score* decreases with a mean value of 4.55%, which is twice the baseline's (2.27%). More specifically, *precision* and *recall* suffer too from twice the degradation, 1.53% and 5.39% respectively.

This degradation of resources strengthen the assumption that the data sets 100, 160, and 200 have a better benchmarking of the dynamics relations, as formulated in the baseline discussion.

Indeed, their corresponding *precision* values diminish less, relatively to the values of the baseline, compared to the other data sets (120, 140 and 180). An average relative diminution of 6% for the former data sets *precision* is observed while the relative diminution of *precision* for the latter data sets is 12.33%.

Put it simply, the resources degradation affects less the data sets with a better benchmarking of the dynamics relations.

9.2.1.3 Random Activation

The last of the synthetic experiments tell us about how much the exploration mechanisms of the data space, through the cooperative behaviors of the framework agents, are essential to a smart and efficient analysis of the analytical links, by reducing their trigger probabilities (cf. 9.1.2).

The experiments results, presented in table 9.3 and displayed in figure 6.8, show that for medium and big data sets (80 to 200) the *f1-score* deteriorates by 26.17% in average, relatively to the baseline. This means that reducing the use of the cooperative exploration mechanisms, firstly, slows down the exploration of the data space, thence avoiding the

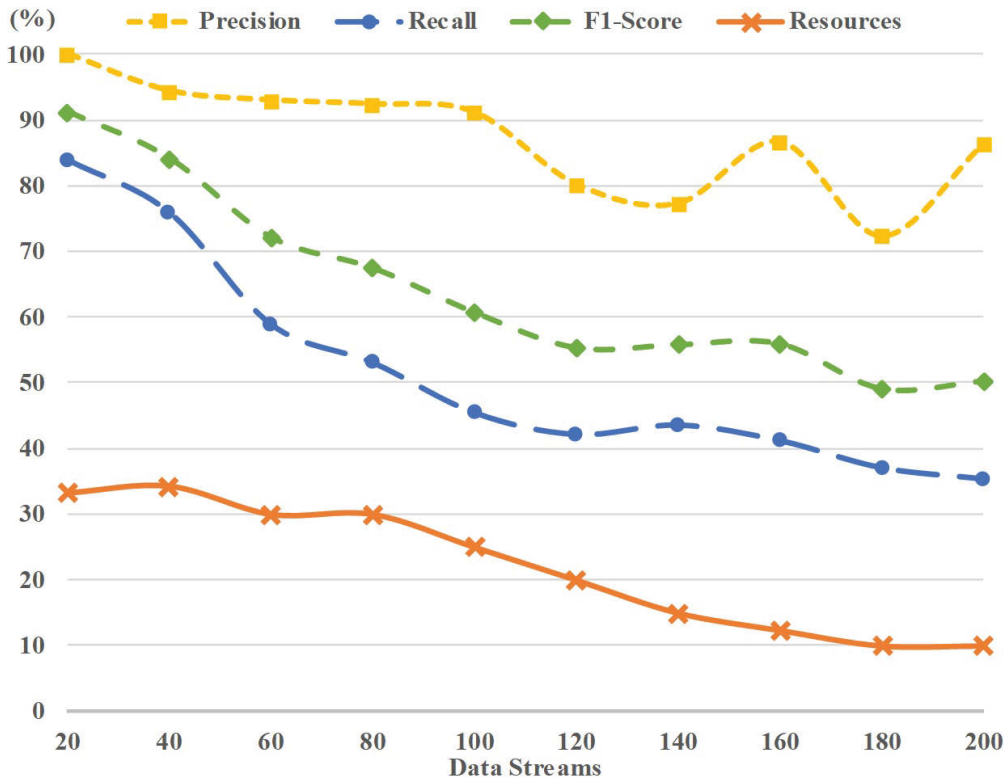

Figure 9.5 — Relevance of the discovered relations with a forced resources degradation (lowest curve).

Data Streams	Resources(%)	Precision(%)	Recall(%)	F1-Score(%)
20	33.26	100	83.96	91.28
40	34.29	94.43	75.98	84.21
60	30	93.01	58.96	72.17
80	30	92.41	53.15	67.49
100	24.99	91.16	45.53	60.73
120	20	80.24	42.24	55.35
140	14.99	77.33	43.67	55.82
160	12.36	86.69	41.31	55.96
180	10	72.38	37.19	49.13
200	10	86.27	35.49	50.29

Table 9.2 — The numeric results of the resources degradation experiments.

analysis of new analytical links for finding new dynamics relations and consequently reducing the *recall*. Secondly, it prevents the analysis of more likely relevant links that may hold a relation.

In addition to the resources degradation management experiments, the assumptions made about the data sets benchmarking quality and the strong relation between *f1-score* and *recall*, in the baseline experiments, is confirmed with the random activation experiments. In other words, the changes occurring on the *recall* have a considerable impact on the changes of the *f1-score*.

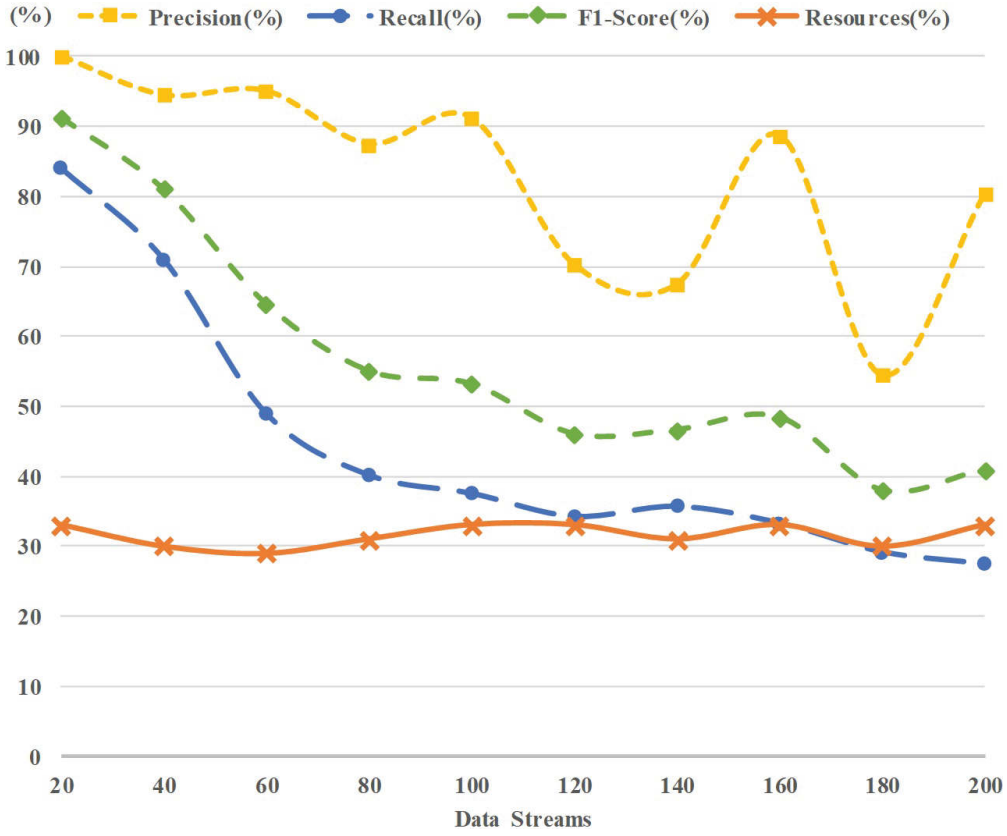

Figure 9.6 — Relevance of the discovered relations with random activation of the exploration mechanisms.

Data Streams	Resources(%)	Precision(%)	Recall(%)	F1-Score(%)
20	33	100	83.96	91.28
40	30	94.43	70.98	81.04
60	29	95.01	48.96	64.62
80	31	87.41	40.15	55.03
100	33	91.16	37.53	53.17
120	33	70.24	34.24	46.04
140	31	67.33	35.67	46.63
160	33	88.69	33.31	48.43
180	30	54.38	29.19	37.99
200	33	80.27	27.49	40.95

Table 9.3 — The numeric results of the random activation experiments.

9.2.2 Data Perturbation: Robustness & Resilience

After studying the output relevance of the framework, now we investigate the evolution of the output relevance through time, with a focus on the ability of the framework to handle perturbations (noisy data).

For this reason, we rely on the data perturbation experiments to evaluate the robustness

of the framework and its resilience. These features are essential for systems interacting with a dynamic environment.

Definition 10. Robustness *[248], or insensitiveness, expresses the ability of a system to endure errors from its environment. The system is said to be robust if it suffers no or few damages (degradation of its performances).*

Definition 11. Resilience *[248], or self-healing, is the capacity of a system to restore itself to a functioning state, after the occurrence of errors or perturbations. The strength of this resilience can be evaluated by measuring the time spent by the system to recover.*

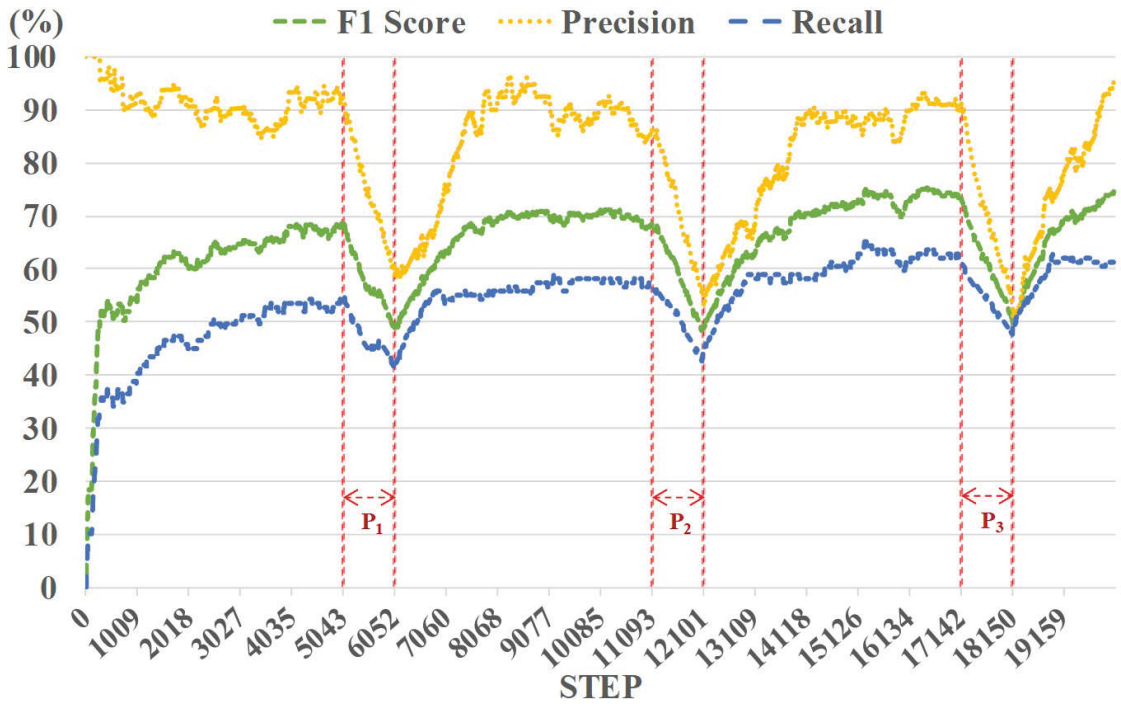

Figure 9.7 — Relevance of the discovered relations with.

As you can see, figure 9.7 displays the progression of the average relevance of the discovered dynamics relations over all the data sets time line. As described previously, the data sets used by the data perturbation experiments include three periods of perturbation: the first period (P_1) goes from 5 040 to 6 048, the second one (P_2) starts at 11 088 and finishes at 12 096, and the third (P_3) lasts from 17 136 to 18 144.

When a perturbation begins, the *precision*, the *recall*, and the *f1-score* of the dynamics relations drops continuously in average by 0.035%, 0.02%, and 0.015% per step respectively. Afterwards, the system starts self-organizing so as to self-adapt to the changes resulted after the occurrence of the perturbations. In other words, self-heal the damages inflicted at the hand of the environment, thence returning to a working state similar to the one before the perturbations. This self-healing capability is exhibited by the increase of the relevance criteria, and even improves the *recall* after a mean time of 500 analytics cycles.

To sum up, even if the robustness of the framework may not be considered good, its resilience though is efficient and makes up for the robustness. It is noteworthy that this

is, at least in part, a desired and engineered result, since a system deployed in a strongly dynamic environment should be responsive to any changes. Therefore, the AMAS4BigData framework is able to carry out the processing of noisy data, which makes it well suited to handle real world data wherein the data contains errors, due to human missteps and/or machines faults and disruptions.

9.2.3 Data Space Exploration

The second category of experiments results focuses on how the AMAS4BigData framework covers the data space (all the possible relations), relying on the *active relations* and *relations coverage* criteria from IoT experiments (cf. 9.1.3).

First, we study the relations exploration of a small data set. Then, we compare it to the exploration of a large data set.

As you can see in figure 9.8, for a small data set, the framework initially covers all the data space quickly, then it restarts the relations exploration from the currently active relations. This cycle of relations coverage is repeated several times during the data set time line, indicating that the framework re-explores all the data space many times, which allows it to detect changes in the data and self-adapt to them.

Moreover, the relations coverage is done with a limited number of computing resources that enable the activation of a proportionally limited number of relations, as exhibited with the *active relations* criterion.

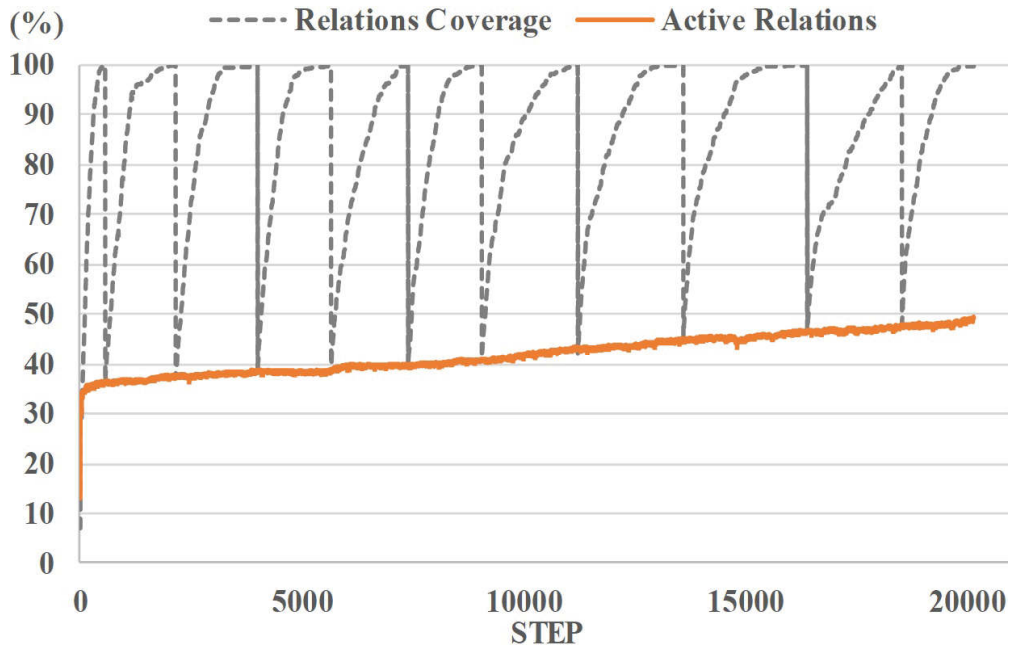

Figure 9.8 — Relations exploration over time in the neOCampus experiment.

Contrary to the neOCampus experiment, the electricity experiment shows (cf. figure 9.9) that the first full coverage of the data space takes a long time, due to the huge size of the data space to be explored (more than 70 000 relations) and the meager amount of available computing resources to do so (0.25% active relations in average).

However, similarly to the neOCampus coverage, the electricity coverage is fully reiterated frequently. This dichotomy of time coverage is explained as follows:

- Initially (from step 0 to 30 000), the system requires a lot of time to build an initial model of all the data space, with a very small pool of resources;
- Then, the system only updates this model which takes less time, because the dynamics relations of the electricity experiment are mostly constant. In other words, the data model remains nearly the same considering the few changes it holds, which is the topic of the next category of experiments results.

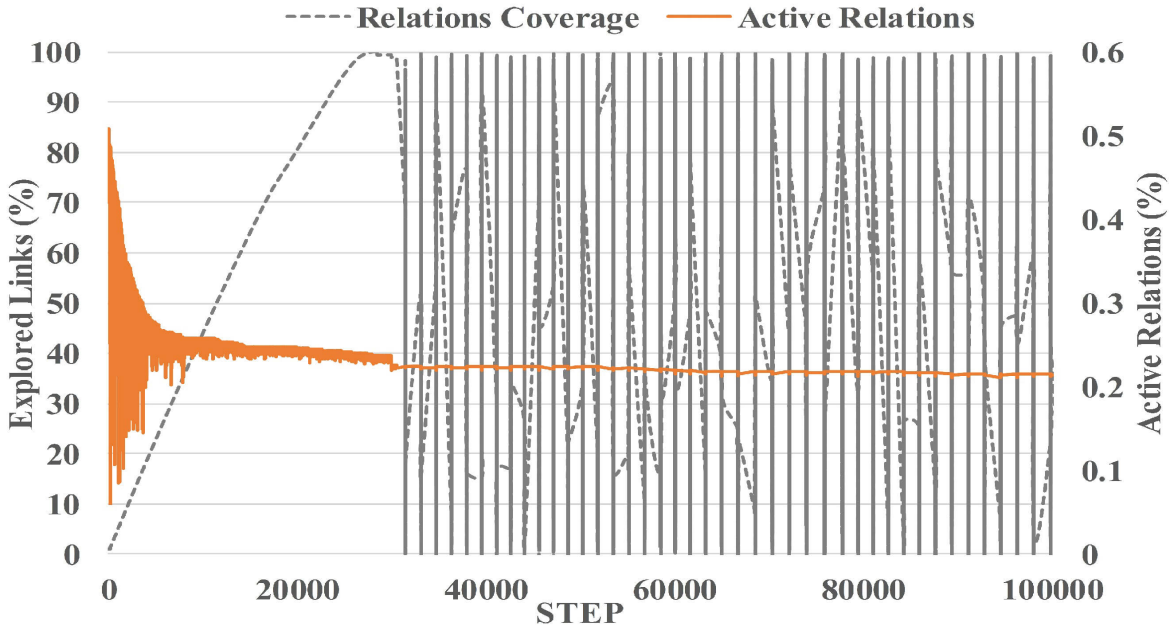

Figure 9.9 — Relations exploration over time in the electricity experiment.

9.2.4 Resources Allocation

This category encompasses the results related to the evolution of the compute agents (the pool of computing resources) for the biggest experiment, the electricity experiment, in order to understand how the AMAS4BigData framework manages its available computing resources. Figure 9.10 indicates the efficiency and usefulness of the resources allocation by the framework.

In the first place we can observe, through the *allocated compute agents* criterion, that all the compute agents are most of the time assigned to analyze a relation, so they are entirely used. Then as shown by the *punctual usefulness*[6] during the initial coverage (step 0 to 30 000), all the compute agents are useful by making modifications to the data model, thanks to their assessment of data changes (presence or absence of dynamics relations).

[6] Discovery of a novelty while analysing a new link.

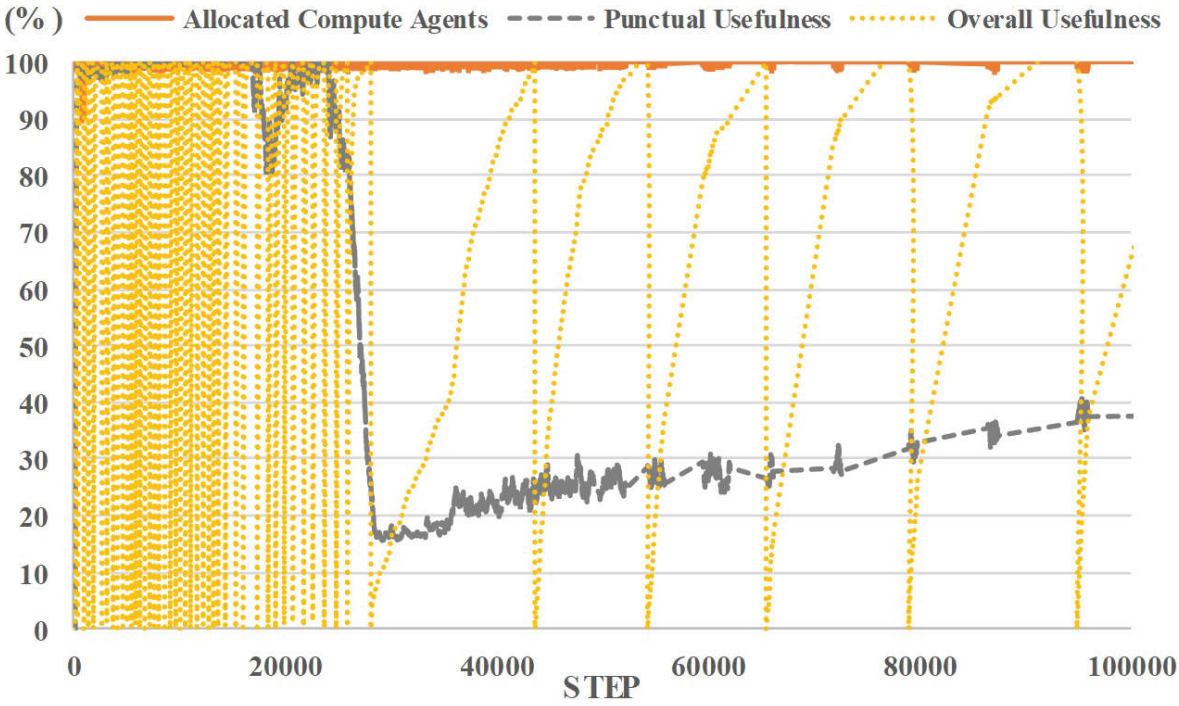

Figure 9.10 — Resources allocation in electricity experiment.

This usefulness of all the compute agents brings the *overall usefulness*[7] to cycle from 0% to 100% very often, which also means that the allocation of computing resources is globally very good. Therefore for the initial phase, the allocation of the compute agents (resources) is efficient and useful. Put it simply, each resource is fully exploited and produces useful information.

After the initial coverage, the *punctual usefulness* of the compute agents drops to 28% and keeps rising to 38% which means now, from step 30 000 until the end, only 28% to 38% of the allocated resources find novelty (appearance or disappearance of a dynamics relation) in the data and update the data model consequently, due to the small amount of changes in the data set (strong consistency). In addition, the mean time before finding changes of the compute agents rises, resulting in longer and less frequent cycles of the *overall usefulness*.

Also since the use of computing resources is not optimal[8] at every step (fluctuacting between 98% and 100%), the framework continuously frees computing resources in order to achieve a full use of the remaining ones, leading to a constant decrease of the quantity of necessary computing resources (the *compute agents count*) as illustrated in figure 9.11. We can note that, when reaching step 50 000 the release of computing resources is slightly slowed down because the use of the available resources is closer to the optimal.

The release of computing resources is very important since it allows other applications to coexist with the AMAS4BigData framework in the same environment, and reduces the framework space-time complexity as explained next.

[7] percentage of agents useful at least once. See also 8.3.2
[8] Optimal use means 100% of the compute agents are allocated every step.

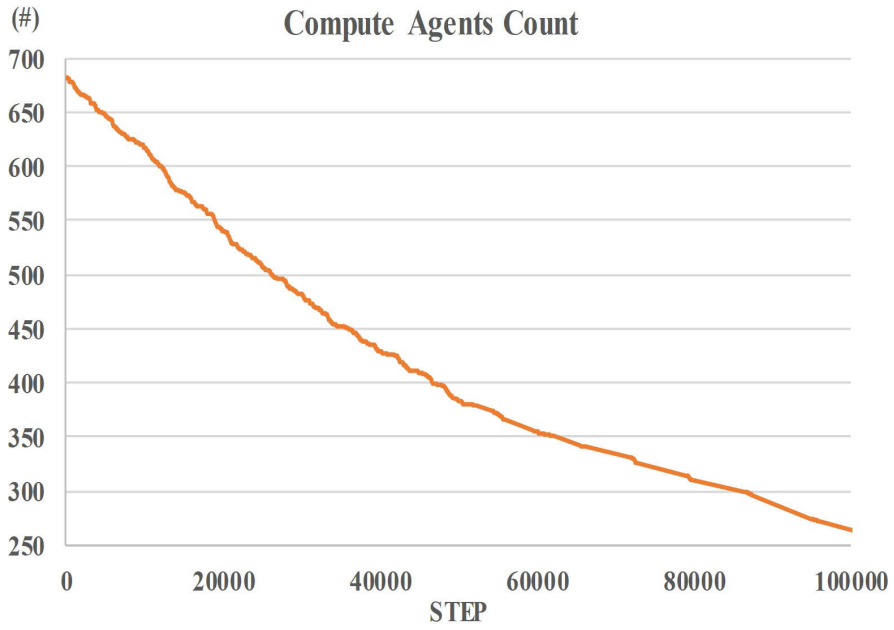

Figure 9.11 — Number of computing resources used/necessary over time in the electricity experiment.

9.2.5 Space-Time Complexity

Processing time and memory usage are two fundamental criteria to judge the ability of a computer system to scale up, especially when dealing with big data because of their high volume and fast generation (cf. 4.1.1).

To study the time complexity and space complexity of the AMAS4BigData framework, this segment of discussion relies on both synthetic experiments, for an endpoint evaluation, and IoT experiments to investigate the progression of processing time, processing speed and memory consumption.

First, figure 9.12 shows the relationships between the amount of computing resources and the space-time complexity, because this is the only factor that impacts on these complexities. As you can see, there is a strong correlation between the number of computing resources and the resulting needs. This is explained by the fact that all the experiments are made on a single machine with a limited number of CPU cores.

So if the framework was provided with a hardware (a machine or a network of machines) of limitless parallel computing cores, it would achieve a constant-like time complexity. However, the overall memory consumption would still be related to the number of compute agents.

In addition, we can observe that even though the data space grows exponentially (see the number of relations in figure 9.1), the framework need of computing resources does not escalate in the same way thanks to its resources allocation policy, formerly discussed in 9.2.4.

Regarding the evolution of the space-time complexities over time, we focus on a midsized data set with the ozone experiment. As illustrated in figure 9.13, the *processing speed (data/second)* of the AMAS4BigData framework is increasing in a logarithmic fashion,

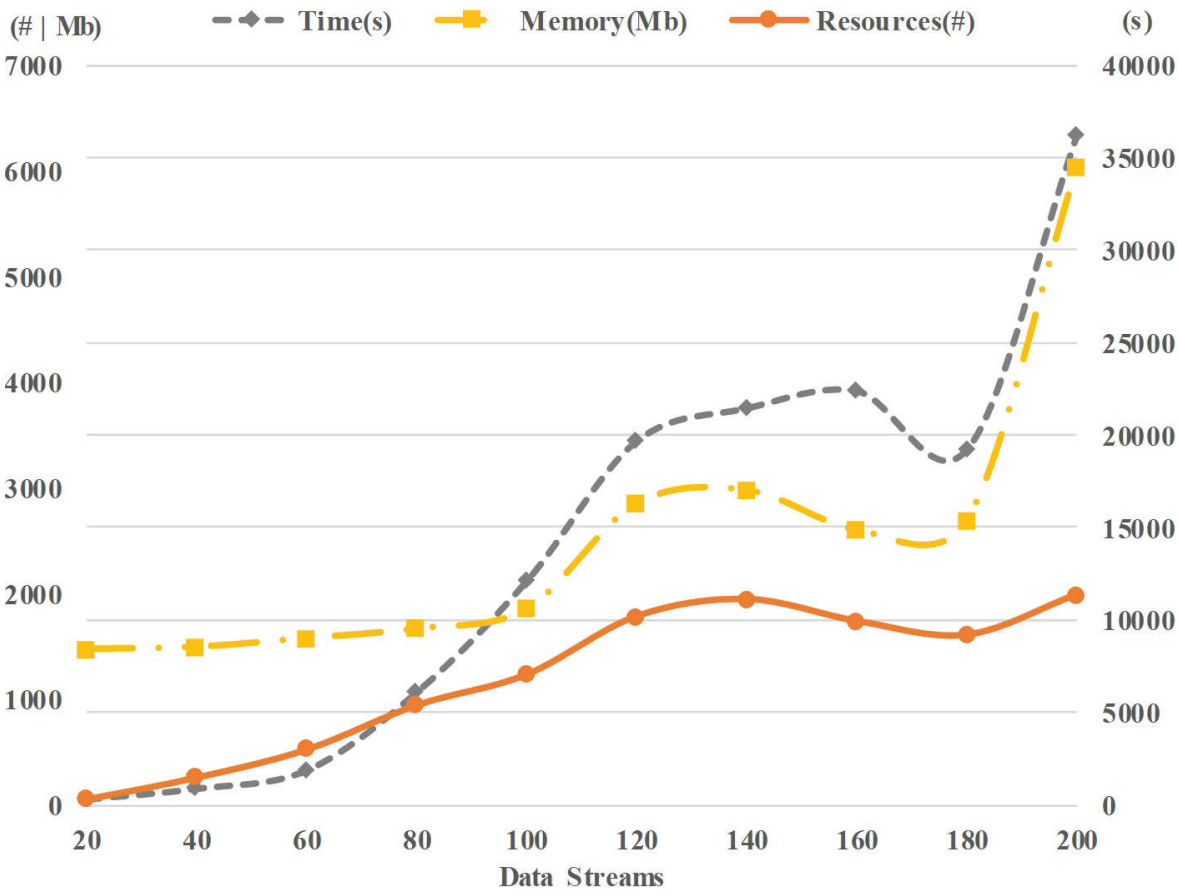

Figure 9.12 — Processing time and memory usage in synthetic experiments.

meaning the quantity of data that the framework processes in one second grows bit by bit. In consequence, the processing time of the data by the framework is nearly logarithmic too. In other words, as the data are coming the framework processes them faster and reduces their overall processing time.

As for the spatial complexity of the framework, like the temporal complexity the tendency of the memory usage is logarithmic as well. Hence, it requires lesser and lesser space proportionally to the data volume processed.

However the actual memory usage, which is growing logarithmically as previously discussed, is cyclic due to one technical feature of the framework implementation. Indeed, the code of AMAS4BigData framework is given in Java 10. This new version of Java introduces a major update of the *Garbage Collector (GC)*[9].

The new *GC* improves the memory management by dividing the available memory into several regions, and efficiently detects when a region is ready to be collected. This distributed memory management is close to the processing paradigm of the AMAS4BigData framework, allowing the *GC* to regularly clear the memory not used anymore by the framework agents.

It should be noted that, even if they are not presented here, the results of the electricity

[9] *The Garbage Collector* reclaims memory occupied by objects that are no longer in use by the program.

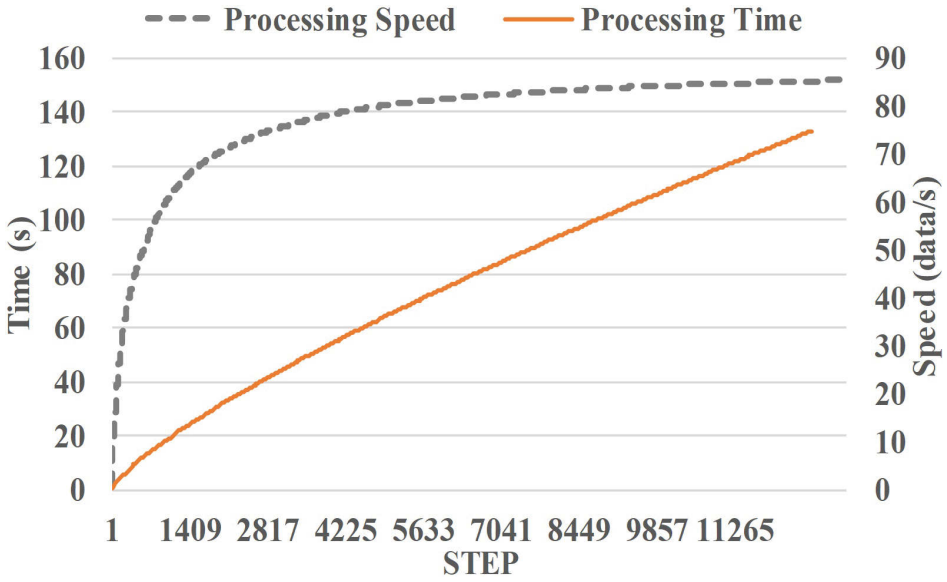

Figure 9.13 — Processing time and Processing Speed over time in ozone experiment.

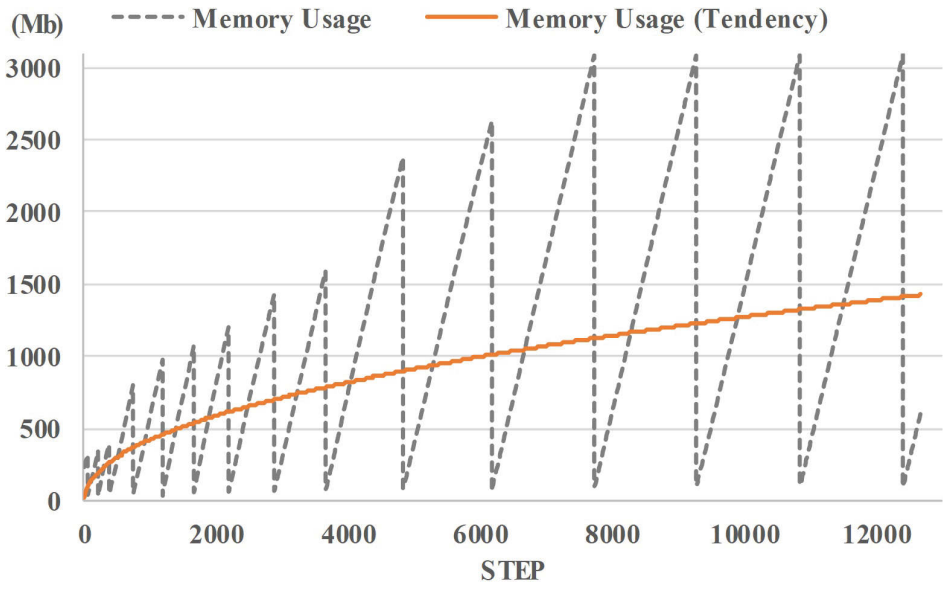

Figure 9.14 — Memory usage over time in ozone experiment.

experiment are consistent with the formerly given analysis.

9.2.6 Anomaly Detection

This last category of results gives us a view on an alternate application of the AMAS4BigData framework: anomaly detection with *DREAM*. We compare its output (anomalies) to the output of *FOD*[10], a conventional anomaly detection technique (cf. 9.1.4).

In this experiment *DREAM* uses the default settings and *FOD* has a contamination factor set to 0.14%. The anomalies detected with both tools are presented in figure 9.15.

[10] *Fourier Outlier Detection (cf. 9.1.4).*

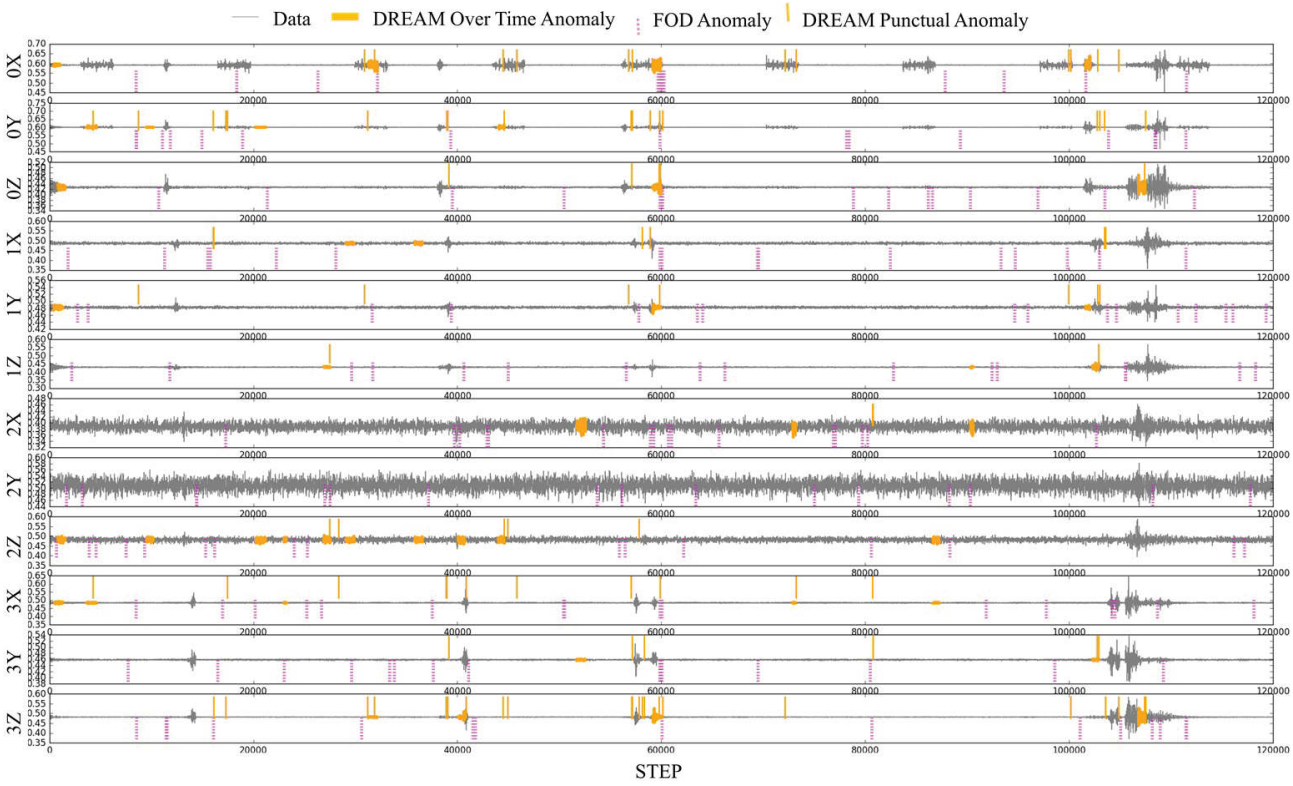

Figure 9.15 — Bridge Anomaly Detection with *DREAM* and *FOD*.

Due to a lack of an oracle[11] that indicates where to find the real anomalies in the data set, we can not assert on which technique gives the better results. However, we can compare *DREAM* and *FOD* to each other:

- ▷ To begin, the *FOD* anomalies tend to be more scattered across the data compared to the anomalies detected by *DREAM*, which may point out to a higher false positive rate of anomaly detection;
- ▷ Besides the punctual anomalies, *DREAM* also detects anomalies that can be observed only over time (cf. 7.4.1.2). To compensate, *FOD* outputs several punctual anomalies close to that time period, as illustrated in figure 9.16 (as seen in the first sub-graph, *"0X"*);
- ▷ Moreover, after an initial adaptation/learning time (20 000 steps), *DREAM* often detects anomalies before *FOD* does (as seen in third sub-graph, *"2X"*).

To conclude this discussion, one must be reminded that unlike *FOD*, *DREAM* doesn't require a heavy extra processing of the data, since the anomaly detection feature is a simple result of its basic functioning (cf. 7.4.1). Furthermore, *DREAM* detects anomalies in real-time while *FOD* needs to processes all the data to extract the anomalies. Therefore, *DREAM* offers a better detection time with less storage requirement.

[11] This is done on the data collected for an ongoing experiment at the *NII* in Japan (cf. 8.2.2).

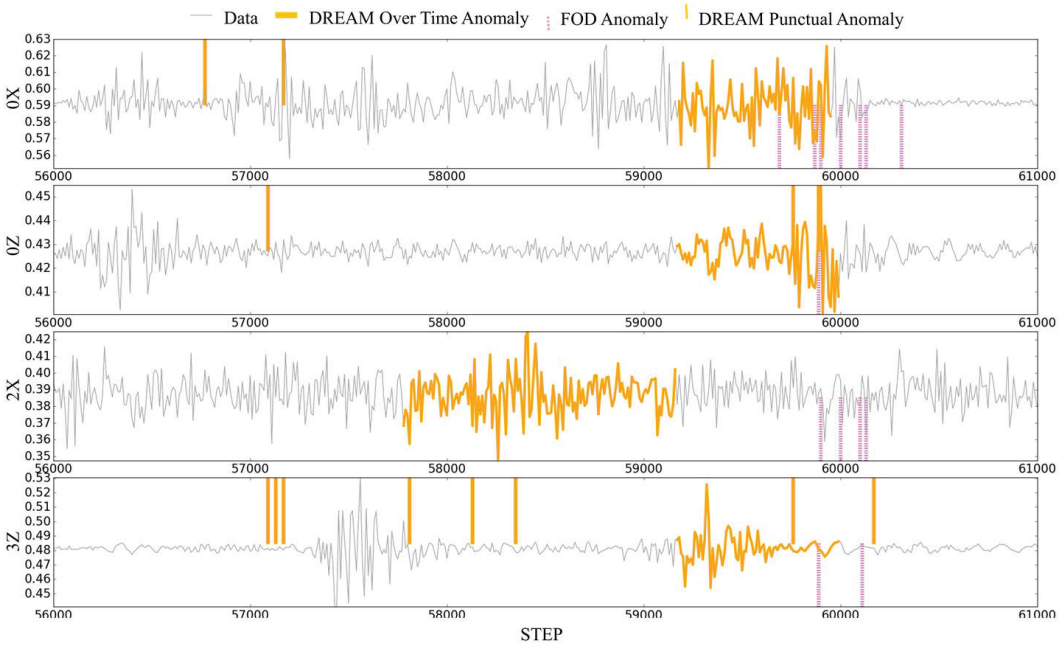

Figure 9.16 — Partial result of the Bridge Anomaly Detection.

9.3 Online Experiment

The experiments presented and described so far dealt with offline stored data, but the AMAS4BigData framework is designed to process data produced in real-time. The experiment presented here shows the ability of the framework to work online, through the use of the application *DREAM* on neOCampus IoT. Figure 9.17 gives a screenshot of the online usage of *DREAM* during an afternoon on the data generated by neOCampus IoT. Each data source is represented by an icon of its type (luminosity, temperature, electricity consumption...) and it is designated following the pattern *building_room_position_ID*.

Indeed, *DREAM* is programmed to also receive and process data streams coming from the network, as explained in 7.3.1. For instance, *DREAM* percept agents are provided with a skill that allows them to receive the sensors data of the neOCampus IoT via the university network. Technically, with this skill *DREAM* agents subscribe themselves to an *MQTT*[12] broker, which sends the data coming from the publishers (the sensors) to their subscribers (the agents).

In addition, DREAM provides a graphical user interface with two main windows. The first one (cf. figure 9.17) displays in real-time the data graph model (cf. 7.2.5) of the input, offline or online, data streams. The second window displays a matrix of the data model and a matrix of detected anomalies, as shown in figure 9.18. It also displays the history of the relations of each data stream and the history of the anomalies as well, like in figure 9.19.

[12] *Message Queuing Telemetry Transport* is an ISO standard *publish-subscribe* based messaging protocol.

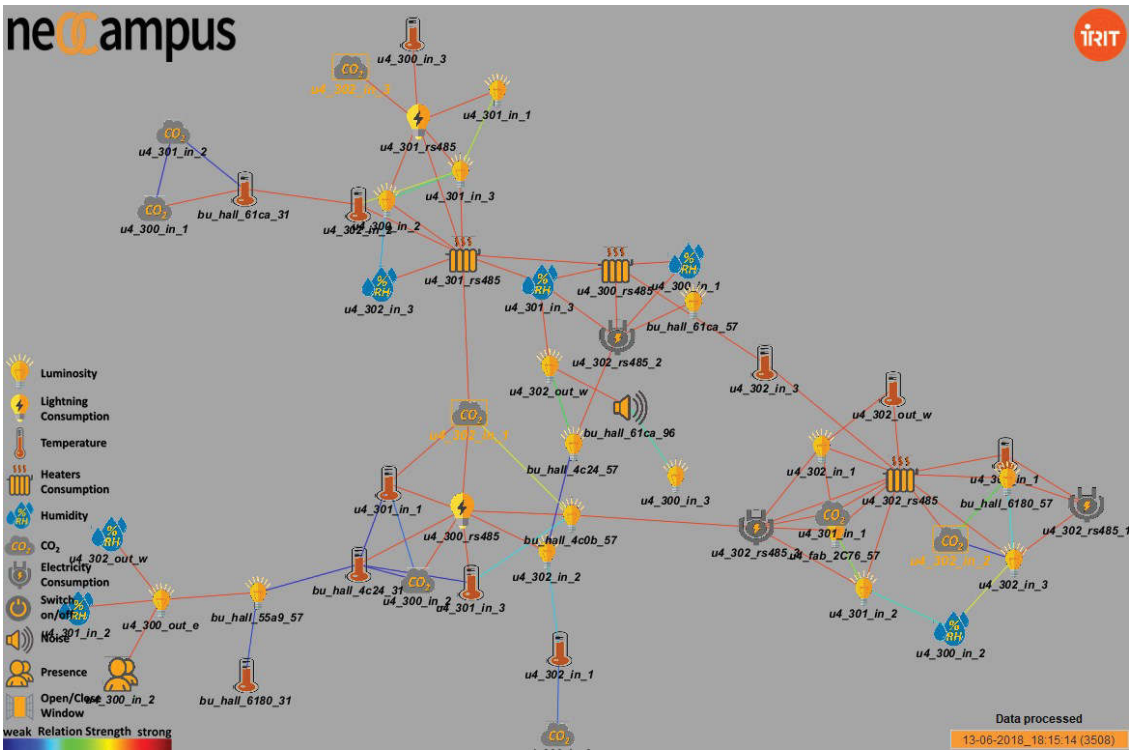

Figure 9.17 — Screenshot of DREAM applied on neOCampus IoT.

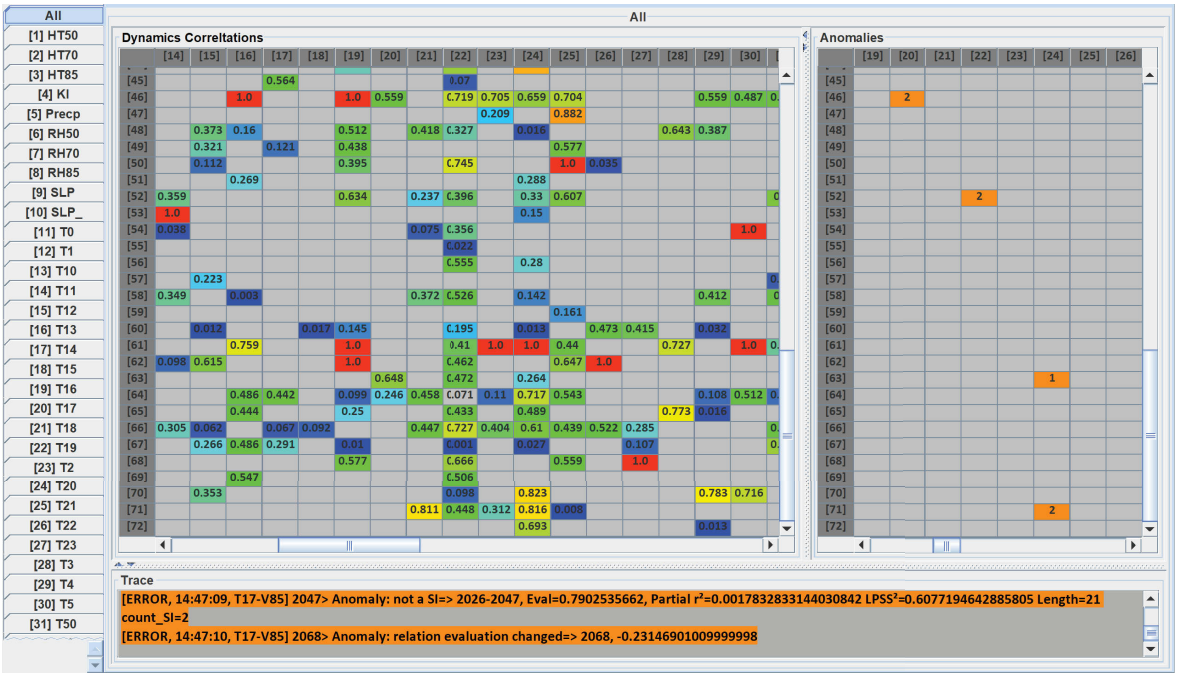

Figure 9.18 — DREAM second window: matrices view.

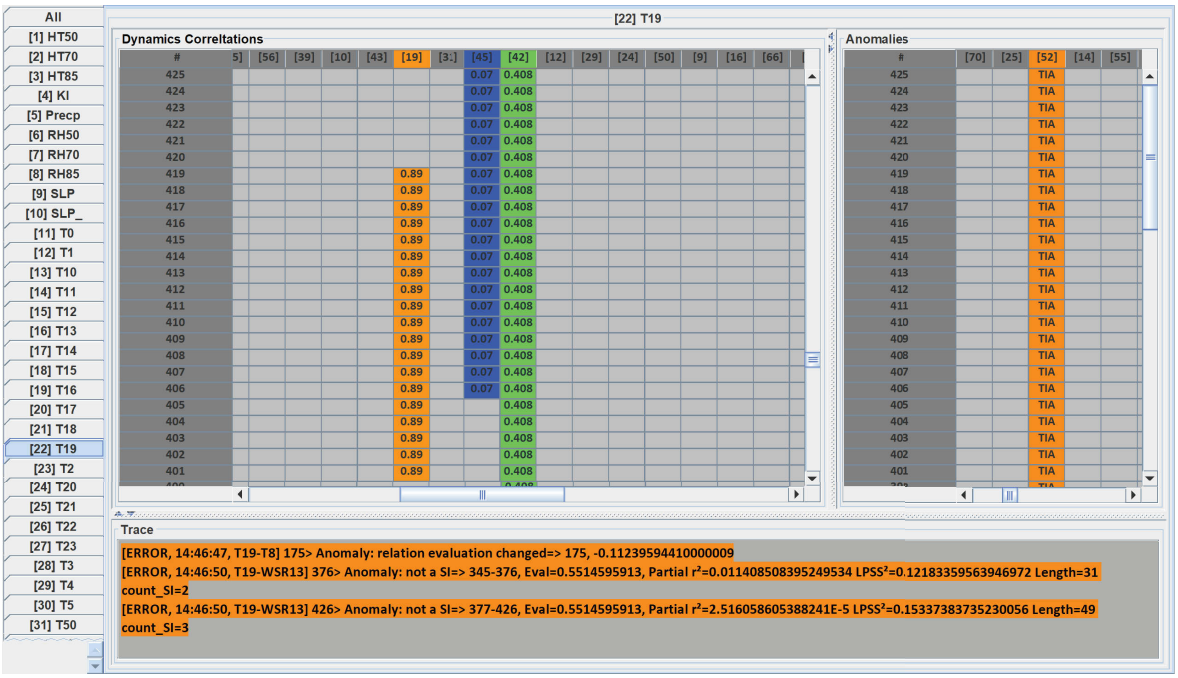

Figure 9.19 — DREAM second window: history view.

9.4 Sum Up: Overview of AMAS4BigData performances

Now that all the experimental results have been presented and discussed, let us gather the individual conclusions in order to draw a broad picture of the AMAS4BigData framework performances with its strengths and weaknesses. The downsides are discussed in second place, since they give hints and directions for future works toward the framework improvement.

9.4.1 AMAS4BigData Strengths

For starter, we can conclude that the *AMAS4BigData* framework scales up rather well from the point of view of output relevance, resources consumption and processing time: the continuous flow of data is efficiently processed thanks to an intelligent use of its computing resources, and the framework adapts itself and updates its strategy of data space exploration, following the evolution of the amount of available computing power, in order to give the best possible results in a reasonable amount of time even in resource limited environments. This complies to the *elasticity* feature stated in section 4.4 as a requirement to meet the rising challenges of modern data analysis.

Also, the framework manages fairly the available resources (computing power and memory), so it does not starve the other applications running in the same environment.

Additionally, thanks to the AMAS technology (cf. 6), the framework can change its internal organisation, so it self-adapts to cope with changes of the data structure (new data streams appearing or data streams disappearing) and recovers from damages caused by disturbances occurring in the data or by the loss of computing power.

Another perk of the framework is its privacy protection feature: unlike conventional analytics frameworks, the *AMAS4BigData* framework hands over the control of the data to their sources (through each percept agent) and gives them a full disclosure of what is done with their data. Therefore, the owner of the data can monitor and control their processing. For example, a person can encrypt and remove any identifiable element from its data, observe what is done with them, and even restrict their access.

9.4.2 AMAS4BigData Weaknesses

Since it is based on a non centralized processing architecture with a lot of agents, the *AMAS4BigData* framework requires a highly parallel computing environment to achieve its full potential, using either one platform containing tremendous CPU cores with shared memory, or using a vast network of machines which is difficult to get for most of people either ways.

The second drawback of the framework is technical too and relates to its coding. Indeed, as described previously (cf. 7.3), the final code is Java based which allows high interoperability[13] of the framework but inhibits optimal use of the computing environment, especially when dealing with HPC platforms.

9.4.3 Optimal Use of AMAS4BigData

From the strengths and weaknesses of the *AMAS4BigData* framework, we can draw a conclusion on how to best use it depending on the nature of the data to be analysed.

If the data are generated in real-time by a network of devices/sensors (IoT), then it is better to deploy the framework directly on the devices (edge computing) and exploit their communication network for agents interaction and movement. However, if the data are offline (stored in a database) then a set up of the framework on an HPC platform (cloud computing) is best suited if the data set is very large, or one can simply use regular machines with limited computing power otherwise.

Furthermore, as intelligent as it can be, the data space exploration by the framework can always be improved by integrating human users feedbacks to guide the system.

[13] Working on different devices with various operating systems.

Conclusion & Perspectives

General Conclusion

WITH the explosion of the quantity and variety of smart devices, and the continuously increasing computing power and networks speed, the big data era entered a new stage wherein the data flood requires a paradigm shift in the way it has been processed because of the new rising data science challenges. Indeed, so far most of the available analytics tools were based on hardware improvement and data distribution.

However, such techniques will eventually reach their break point, since they are not tailored to handle the complex nature of the data flood. For this reason, this work aimed to design a new big data analytics framework, called *"AMAS4BigData"*, that can deal in real-time with the data generated by complex systems using a bio-inspired collective artificial intelligence, the *self-Adaptive Multi-Agent Systems (AMAS)*.

The framework development process started with studying the origins and the current state of the data science, as done so far, with its ongoing challenges. Then, I defined new challenges (rising challenges) expressing new problems that emerges when dealing with data of complex systems. They also give the new features that must be exhibited by analytics systems in order to achieve the evolution of the data science. The second step of the framework development, consisted in extensively surveying the state-of-the-art analytics tools, extracting their features and examining their architectures, in order to get an overview of the current data and big data analytics. Then, identifying their inadequacies to the complex systems data and overcoming them by means of the *AMAS* technology.

In concrete terms, I broke the conventional rigid analytics pipeline into several smaller pipelines, dedicated to one data source, in the shape of autonomous agents that use local data mining and cooperate thanks to their unique behaviors, for the sake of an efficient and smart data analysis. As a result, the *AMAS4BigData* framework exhibits useful features (scalability, elasticity, genericity, resilience, privacy protection), to deal with the complex data flood.

Moreover, the framework was implemented in a concrete application designated as *"Dynamic data Relation Extraction using Adaptive Multi-agent systems (DREAM)"*, which builds a model of the data relationships based on their evolution similarity, measured by a newly designed analytical tool called *"Dynamics Correlation"*. This new data analytics system is especially useful to find rare and scattered information from a huge amount of data, like

identifying the relevant inputs to be thoroughly evaluated by context learning systems [258].

From a technical perspective, the *AMAS4BigData* framework was coded with *SARL*, an agent oriented programming language that provides fundamental abstractions for easy and practical implementation of modern complex software applications with a simple yet powerful syntax (cf. 7.3.2). It results in a big *SARL* project, especially when considering the number of agents used to analyse data (more than 10 000 for the largest data set), and a very small one compared to conventional data analytics framework due to the small number of lines of code of the *AMAS4BigData* framework.

DREAM enables the experimental evaluation the *AMAS4BigData* framework with several criteria under various situations using different data sets. The experimental results show that *DREAM* outputs are relevant, since it discovers most of the true dynamics relations between the data, under normal circumstances (exploration mechanisms always applied and automatic computing resource management), and it gracefully decreases when dealing with bigger data sets. Furthermore, the experiments also demonstrate that the outputs relevance deteriorates when the exploration mechanisms are restrained or the available computing power is restricted.

As for the framework scaling up capability, the experimental results show: on one hand when the number of input data streams is constant, the time and space requirements of the framework to process the incoming data flow is better than linear (logarithmic). On the other hand, when the framework receives more and more data sources, the size of the data space to be explored grows exponentially ($O(n^2)$) ,which should escalate the computing resources requirements of the framework but it does not. Instead it adjusts its functioning according to the available computing power and gives up some of its performances, in order to be able to process the new data with a linear amount of computations.

Additionally, as its name indicates it, the *AMAS4BigData* framework rely on the *AMAS* theory which is built around the cooperation paradigm. Thence, it constitutes the core of this new dynamic big data analytics framework.

Thanks to the agents cooperation, I was able to design and implement a set of tools that shift the big data analytics paradigm from *"the more data we get and process the better are the results"* to*"the smarter and the more efficiently we explore the data the better and faster are the results"*, by defining interaction rules inside the agents (bottom-up approach) that allow them to help each others when they can (neither selfish nor altruistic) in order to split fairly the data analysis workload. More specifically, due to:

- the exploration mechanisms (cf. 6.2.2.3), the agents help each others to pinpoint both the areas (link) that are better to visit, with the *triangulation* mechanism for example, and the ares that agents should leave by cutting the links that have produced no information, like the split phase from the *split-linking mechanism*;
- and the criticality function (cf. 6.3.4), which dictates which percepts have more priority to get a computing resource. Therefore, each percept computes its own criticality and the compute agents choose the ones with the highest value (the most critical) while avoiding any starvation problem, hence the system tries to balance between *exploration* (looking for new links) and *exploitation* (producing information from the current links).

Consequently, the system moves through the data space and explores only the areas (nodes and links) that are most likely holding new useful information.

Another kind of cooperation should be mentioned: the *Human-Machine* cooperation loop. On one hand the system (the machine) carries out all the heavy low level computation and presents its results to its human user. On the other hand, the user can think on a higher level of these results and send back to the system his guidance on the data exploration by means of feedbacks.

For instance in the application *DREAM*, the user can designate a set of links to be analysed first and a set of links to avoid. Then, the system integrates these feedbacks and focuses on the region of the data space indicated by the user. This cycle is repeated in order to find new information and thus discovering new knowledge more efficiently (faster and more relevant).

In conclusion, even if there is no formal proof, yet, that the cooperative exploration of a data space leads to the right results accurately, the experimental evaluation demonstrated that *DREAM* finds most off all the dynamics correlations efficiently.

Another interesting and powerful feature of the *Self-Adaptive Multi-Agent Systems* is the *self-observation*, meaning that a system can monitor itself, judge how well its current organisation is adapted to thrive in its environment, in order to change this organisation through reconfigurations that aims to improve its functioning within the environment. Thus, the self-observation is another condition for the self-adaptation of AMAS systems.

In the *AMAS4BigData* framework, the self-observation is achieved through its agents criticality periodic computation. Indeed at the end of each analytics cycle, the system examines how it did handle the incoming data streams and infers which data streams it must focus on for the next cycle, according to the available computing power for the sake of better knowledge production. Finally, as an autonomous process, this self-observation prevents the user of this system to have to reset its parameter to get the right results.

Perspectives

Even though the *AMAS4BigData* framework is ready to use, and was successfully implemented, its design and realisation have room for improvement as future work. These potential improvements are divided in theoretical and technical perspectives.

Regarding its internal architecture, the *AMAS4BigData* framework can be extended with agents behaviors to increase the efficiency of the data space exploration. Also, the application *DREAM* can benefit from an additional reasoning module that takes the discovered dynamics relations to infer even more complex relations, like causal relations and multi-party relations that improve the produced knowledge;

As for the technical perspective, we can optimize the code of the framework, so it overcomes its weaknesses (cf. 9.4) and be optimally used on any environment, especially the highly parallel ones. Another perspective is the implementation of the prospective tools based on *AMAS4BigData*, presented in 7.5 (data clustering, horizontal and vertical clustering, and frequent item-sets mining).

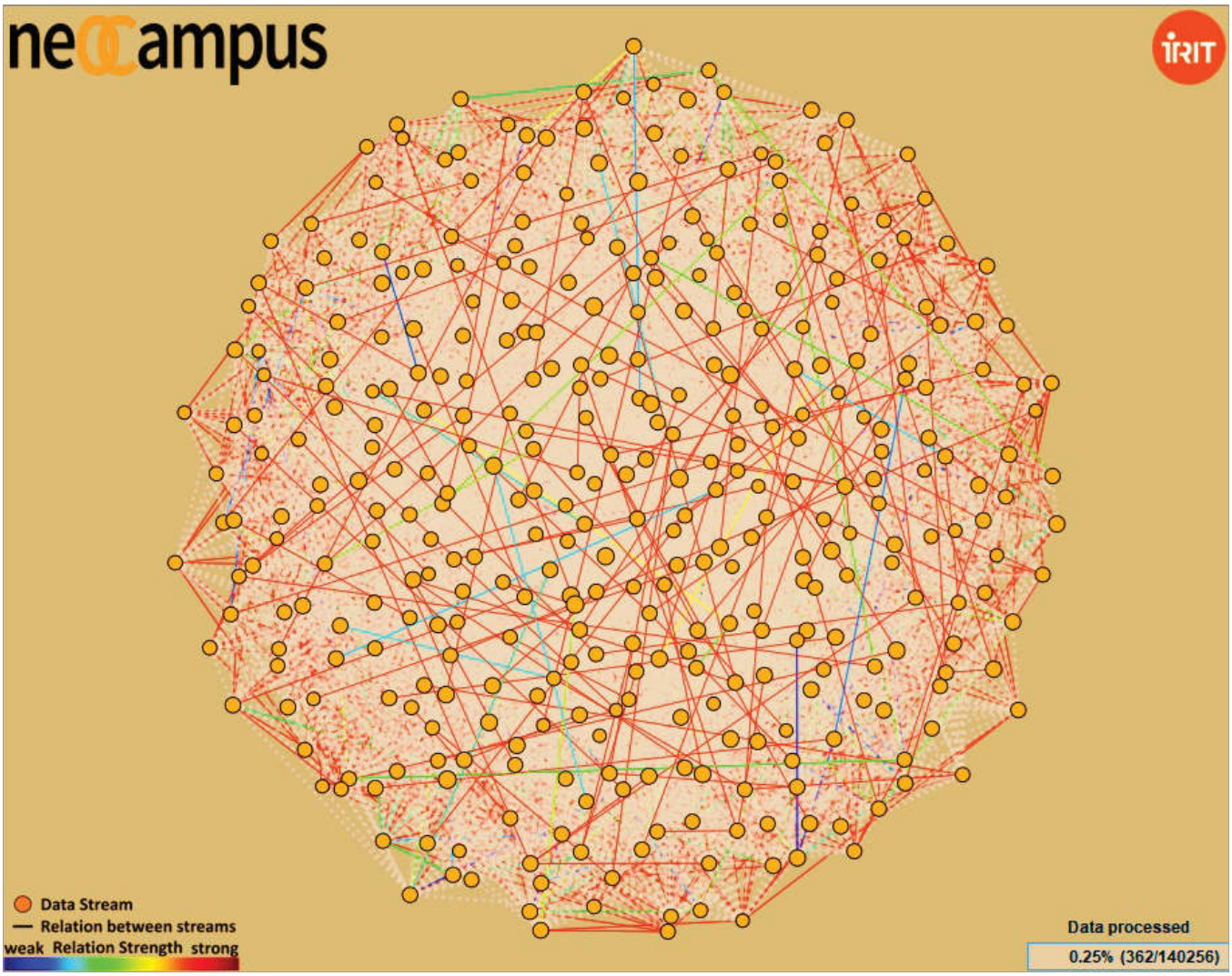

Figure 9.20 — Example of DREAM application.

AMAS4BigData
Adaptive Multi-Agent Systems
for Dynamic Big Data Analytics

Appendix

SARL: Statically-typed Agent-pRogramming Language

1. Definition & Origin

SARL [257] is a statically-typed agent-programming language that aims at providing the fundamental abstractions for dealing with *concurrency*, *distribution*, *interaction*, *decentralization*, *reactivity*, *autonomy* and *dynamic reconfiguration*.

These high-level features are now considered as the major requirements for an easy and practical implementation of modern complex software applications. The agent-oriented paradigm holds the keys to effectively meet this challenge.

Syntactically and semantically SARL has its roots in the Java programming language but improves on many aspects [256]:

- **Agent specific statements**: provide specific statements for agent programming;
- **Type inference**: you rarely need to write down type signatures anymore;
- **Lambda expressions**: concise syntax for anonymous function literals;
- **Operator overloading**: make your libraries even more expressive;
- **Extension methods**: enhance closed types with new functionality;
- **Powerful switch expressions**: type based switching with implicit casts;
- **No statements**: everything is an expression;
- **Full support for Java generics**: including all conformance and conversion rules;
- **Translates to Java not bytecode**: understand what is going on and use your code for platforms such as Android or GWT.

Unlike other JVM languages, SARL has zero interoperability issues with Java: everything you write interacts with Java exactly as expected, and SARL methods can be called from Java, too, in a completely transparent way. At the same time, SARL is much more concise, readable and expressive.

2. Standard Compilation Process

The SARL tool-chain is the set of programming tools that are used to perform a multi-agent system with SARL. As illustrated in figure 21, three types of tools are used in sequence in order to create and run an agent-based system:

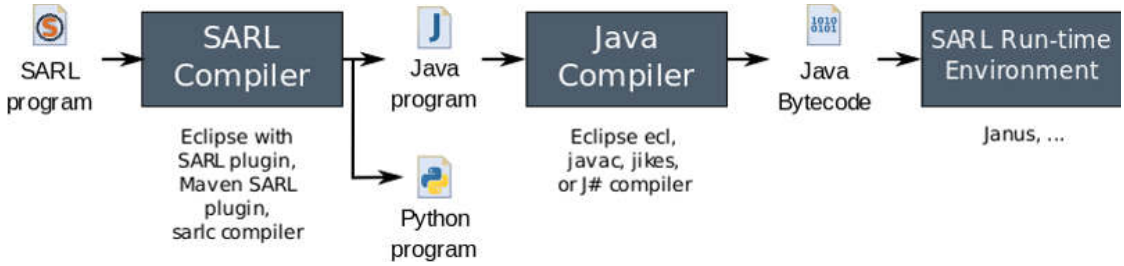

Figure 21 — Compilation process of SARL based applications.

- **SARL Compiler**: the SARL compiler transforms the source SARL language into the target language. Because most of the agent frameworks are written in Java, the SARL compiler targets this object-oriented programming language by default. The SARL compiler translates SARL statements into their object-oriented equivalent statements.

- **Java Compiler**: the files generated by the SARL compiler must be compiled. The result from the Java compilation is a collection of binary files (byte-code files) that may be run by a virtual machine.

- **SARL Run-time Environment**: the SARL Run-time Environment (SRE) is a collection of tools enables running an agent-based application written with SARL. Such an SRE must implement each service and feature that are assumed to be provided by the run-time environment. When the target platform is Java-based, the SRE is composed by a standard Java Runtime Environment (JRE), and the Janus Java library, which provides the base classes for running agents written with SARL.

3. Agents Identification & Location

When it is spawn, an agent is placed inside the system in a place named *"Context"*. Contexts defines the boundary of a sub-system, and gathers a collection of *"spaces"*. Spaces is the support of the interaction between agents respecting the rules defined in the spaces' specification.

All agents are incorporated into the *Default Context* and during their lifetime, they may join and participate in other contexts that are not the default context. They are called *"the external contexts"*. In each context, there is at least one particular space called *Default Space* to which all agents in this context belong. Thus, it ensures the existence of a common shared space to all agents in the same context. In addition, Agents can create specific public or private spaces to achieve their personal goals.

Moreover, agents have a unique identifier *"UUID"* (Unique Universal IDentifier) for all the multi-agent system and an *"address"* for each space they belong to.

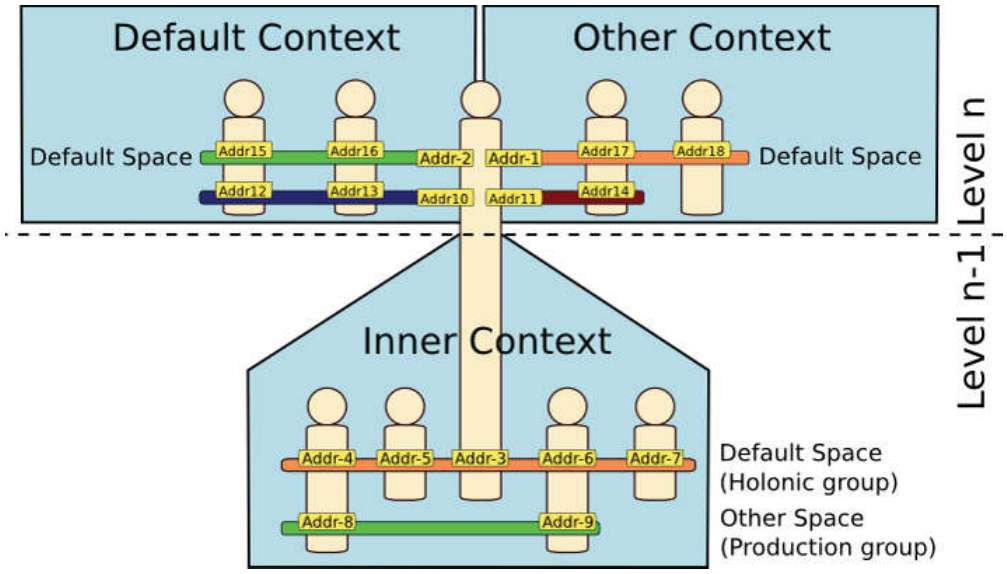

Figure 22 — Agents Identification & Location.

4. Architecture Components & Meta Model

Before detailing the architecture and the definition tools of an agent, it may be helpful to understand SARL components, illustrated in figure 23:

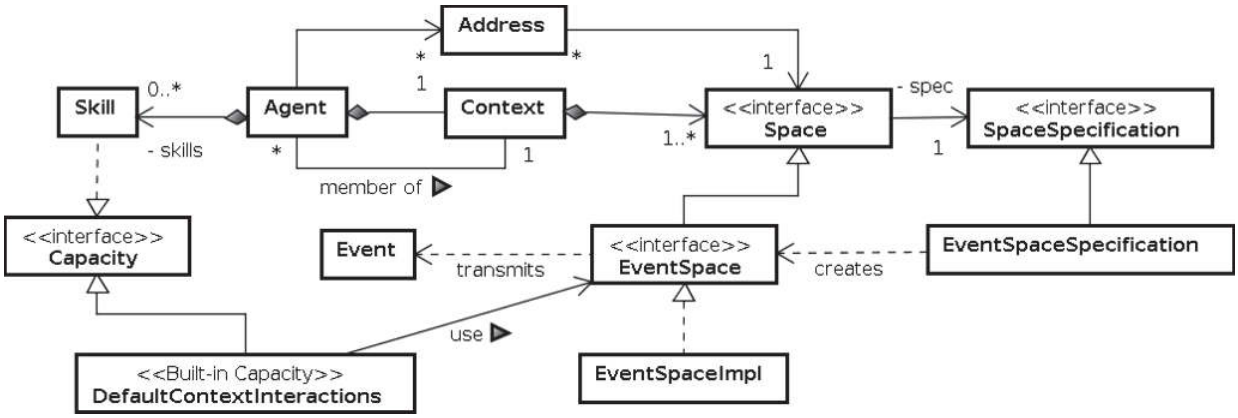

Figure 23 — SARL meta model.

- **Action**: code (a public method or function) that transforms a part of the designed system or its environment. This transformation guarantees resulting properties if the system before the transformation satisfies a set of constraints. An Action is defined in terms of pre- and post-conditions.
- **Behavior**: maps a collection of perceptions represented by Events to a sequence of

Actions. The various behaviors of an agent communicate using an event-driven approach. This Behavior may be used by an agent for building its global behavior.

- **Event**: a data structure composed of attributes and exchanged among the agents or the behavioral units of agents, inside a given Space.
- **Capacity**: specification of a collection of Actions. This specification makes no assumptions about the implementation of each Action. It is used to specify what an Agent can do, what behavior is required for its execution. Consequently, only Action signatures can be defined inside a Capacity.
- **Skill**: implementation of a Capacity. In other words, it is a collection of Actions implementing a Capacity as described in this specification.

5. Agent Architecture

The architecture of an agent is illustrated by figure 24. To sum up, an agent has a global Behavior defined by several Behaviors. Those behaviors might relies on various Skills, according to the related Capacities, in order to execute some Actions following the perception of an Event.

Also, agents have a set of built-in capacities essential to respect the commonly accepted competencies of agents, such *autonomy*, *reactivity*, *pro-activity* and social capacities

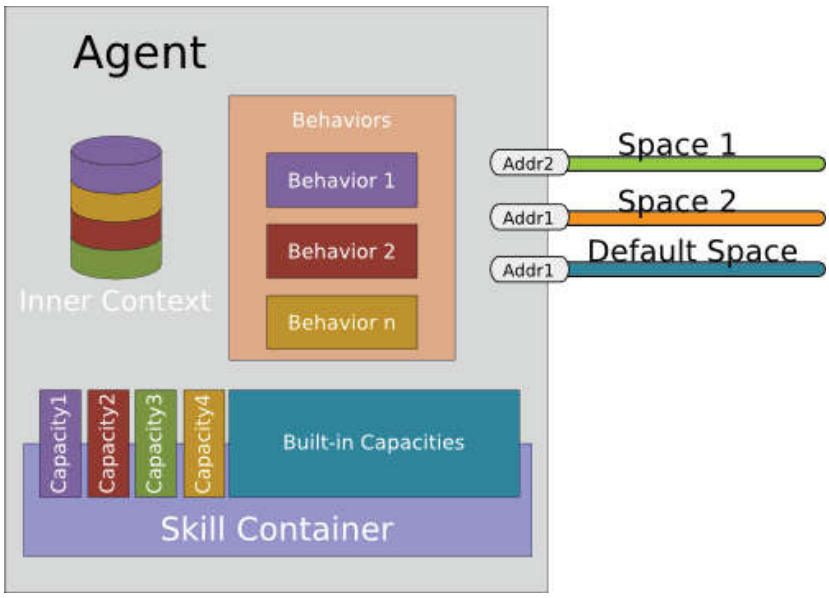

Figure 24 — Agent Architecture.

Furthermore, An Agent can dynamically evolve by acquiring (learning) new Capacities, and it can also dynamically change the Skill associated with a given Capacity. Acquiring a new Capacity enables an Agent to get access to new behaviors. This provides Agents with a self-adaptation mechanism that allows them to dynamically change their architecture according to their current needs and goals.

Bibliography

[1] E. A. Lee, "Cyber physical systems: design challenges", in *11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, IEEE, 2008, pp. 363–369.

[2] W. S. Cleveland, "Data science: an action plan for expanding the technical areas of the field of statistics", *International statistical review*, vol. 69, no. 1, pp. 21–26, 2001.

[3] *CODATA, The Committee on Data for Science and Technology*, 2002. [Online]. Available: `http://www.codata.org/` (visited on 11/07/2016).

[4] *Data Science Journal Volume 1, Issue 1. Retrieved from Japan Science and Technology Information Aggregator. April 2002*, 2002. [Online]. Available: `https://www.jstage.jst.go.jp/browse/dsj/1/0/_contents/-char/en` (visited on 11/07/2016).

[5] *Journal of Data Science: Volume 1, Number 1, January 2003*, 2003. [Online]. Available: `http://www.jds-online.com/v1-1` (visited on 11/07/2016).

[6] *data, n., OED*, en-GB, 1960. [Online]. Available: `http://www.oed.com/view/Entry/296948` (visited on 07/12/2017).

[7] M. Cox and D. Ellsworth, "Application-controlled demand paging for out-of-core visualization", in *Visualization'97., Proceedings*, IEEE, 1997, pp. 235–244.

[8] D. Laney, "3d data management: controlling data volume, velocity and variety", *META group research note*, vol. 6, no. 70, p. 1, 2001.

[9] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact", en, vol. 36, no. 4, p. 24, 2012.

[10] S. Negash, "Business intelligence", *Communications of the association for information systems*, vol. 13, no. 1, p. 15, 2004.

[11] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases", *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[12] D. Simberloff, B. Barish, K. Droegemeier, D. Etter, N. Fedoroff, K. Ford, L. Lanzerotti, A. Leshner, J. Lubchenco, M. Rossmann, *et al.*, "Long-lived digital data collections: enabling research and education in the 21st century", *National Science Foundation*, 2005.

[13] G. Press, "A very short history of data science", *Forbes. com*, 2013.

[14] J. W. Tukey, "The future of data analysis", *The annals of mathematical statistics*, vol. 33, no. 1, pp. 1–67, 1962.

[15] P. Naur, "Concise survey of computer methods", 1974.

[16] *IASC-ISI | International Association for Statistical Computing*, en-US, 1977. [Online]. Available: `http://iasc-isi.org/` (visited on 12/11/2016).

[17] J. W. Tukey, *Exploratory data analysis*. Reading, Mass., 1977, vol. 2.

[18] G. Piatetsky-Shapiro, "Knowledge discovery in real databases: a report on the ijcai-89 workshop", *AI magazine*, vol. 11, no. 4, p. 68, 1990.

[19] J. Berry, "Database marketing", *Business Week*, vol. 5, pp. 34–40, 1994.

[20] C. Hayashi, K. Yajima, H. H. Bock, N. Ohsumi, Y. Tanaka, and Y. Baba, *Data Science, Classification, and Related Methods: Proceedings of the Fifth Conference of the International Federation of Classification Societies (IFCS-96), Kobe, Japan, March 27–30, 1996*. Springer Science & Business Media, 2013.

[21] A. Swan and S. Brown, "The skills, role and career structure of data scientists and curators. an assessment of current practice and future needs. report to the joint information systems committee (jisc)", *Truro: Key Perspectives for JISC. Retrieved February*, vol. 15, p. 2012, 2008.

[22] I. W. G. on Digital Data, "Harnessing the power of digital data for science and society", *Report to the Committee on Science of the National Science and Technology Council*, 2009.

[23] K. Cukier, *Data, data everywhere: A special report on managing information*. Economist Newspaper, 2010.

[24] A. Szalay, "Extreme data-intensive scientific computing", *Computing in Science & Engineering*, vol. 13, no. 6, pp. 34–41, 2011.

[25] R. E. Bryant, "Data-intensive scalable computing for scientific applications", *Computing in Science & Engineering*, vol. 13, no. 6, pp. 25–33, 2011.

[26] A. Lesk, *Introduction to bioinformatics*. Oxford University Press, 2014.

[27] J. McDermott, R. Samudrala, R. Bumgarner, K. Montgomery, and R. Ireton, *Computational systems biology*. Springer, 2009.

[28] F.-Y. Wang, K. M. Carley, D. Zeng, and W. Mao, "Social computing: from social informatics to social intelligence", *IEEE Intelligent systems*, vol. 22, no. 2, 2007.

[29] G. Brumfiel, "High-energy physics: down the petabyte highway", *Nature News*, vol. 469, no. 7330, pp. 282–283, 2011.

[30] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: the next frontier for innovation, competition, and productivity", 2011.

[31] J. Archenaa and E. M. Anita, "A survey of big data analytics in healthcare and government", *Procedia Computer Science*, vol. 50, pp. 408–413, 2015.

[32] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant, "Detecting influenza epidemics using search engine query data", *Nature*, vol. 457, no. 7232, p. 1012, 2009.

[33] B. Kayyali, D. Knott, and S. Van Kuiken, "The big-data revolution in us health care: accelerating value and innovation", *Mc Kinsey & Company*, vol. 2, no. 8, pp. 1–13, 2013.

[34] S. Allwinkle and P. Cruickshank, "Creating smart-er cities: an overview", *Journal of urban technology*, vol. 18, no. 2, pp. 1–16, 2011.

[35] G. P. Hancke, G. P. Hancke Jr, *et al.*, "The role of advanced sensing in smart cities", *Sensors*, vol. 13, no. 1, pp. 393–425, 2012.

[36] J. Wang, Y. Wu, N. Yen, S. Guo, and Z. Cheng, "Big data analytics for emergency communication networks: a survey", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1758–1778, 2016.

[37] M. Dodge and R. Kitchin, "The automatic management of drivers and driving spaces", *Geoforum*, vol. 38, no. 2, pp. 264–275, 2007.

[38] D. Sui, S. Elwood, and M. Goodchild, *Crowdsourcing geographic knowledge: volunteered geographic information (VGI) in theory and practice*. Springer Science & Business Media, 2012.

[39] A. Medvedev, P. Fedchenkov, A. Zaslavsky, T. Anagnostopoulos, and S. Khoruzhnikov, "Waste management as an iot-enabled service in smart cities", in *Conference on smart spaces*, Springer, 2015, pp. 104–115.

[40] P. Piché, E. Belghache, N. Verstaevel, and B. Lartigue, "Impact of eco-feedback on the behavior of campus users", 2017.

[41] Y. Song, G. Zhou, and Y. Zhu, "Present status and challenges of big data processing in smart grid", *Power System Technology*, vol. 37, no. 4, pp. 927–935, 2013.

[42] A. Kylili and P. A. Fokaides, "European smart cities: the role of zero energy buildings", *Sustainable Cities and Society*, vol. 15, pp. 86–95, 2015.

[43] N. Singer, "Mission control, built for cities", *New York Times*, vol. 3, 2012.

[44] M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Manyika, "Big data: the next frontier for innovation, competition, and productivity", *McKinsey Global Institute*, 2011.

[45] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data", *Information Sciences*, vol. 275, pp. 314–347, 2014.

[46] J. Groshek and A. Al-Rawi, "Public sentiment and critical framing in social media content during the 2012 us presidential campaign", *Social Science Computer Review*, vol. 31, no. 5, pp. 563–576, 2013.

[47] J. A. Johnson, "From open data to information justice", *Ethics and Information Technology*, vol. 16, no. 4, pp. 263–274, 2014.

[48] J. Vlahos, "The department of pre-crime", *Scientific American*, vol. 306, no. 1, pp. 62–67, 2012.

[49] D. Hill, "On the smart city: or, a 'manifesto' for smart citizens instead", *City of Sound*, vol. 1, 2013.

[50] A. Acquisti and R. Gross, "Predicting social security numbers from public data", *Proceedings of the National academy of sciences*, pnas–0 904 891 106, 2009.

[51] K. Crawford and J. Schultz, "Big data and due process: toward a framework to redress predictive privacy harms", *BCL Rev.*, vol. 55, p. 93, 2014.

[52] R. J. Brachman and T. Anand, "The process of knowledge discovery in databases", in *Advances in knowledge discovery and data mining*, American Association for Artificial Intelligence, 1996, pp. 37–57.

[53] R. L. Ackoff, "From data to wisdom", *Journal of applied systems analysis*, vol. 16, no. 1, pp. 3–9, 1989.

[54] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[55] E. Mjolsness and D. DeCoste, "Machine learning for science: state of the art and future prospects", *science*, vol. 293, no. 5537, pp. 2051–2055, 2001.

[56] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey", *Journal of Big Data*, vol. 2, no. 1, pp. 1–32, 2015.

[57] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Springer, 2015. DOI: 10.1007/978-3-319-10247-4.

[58] H. Garcia-Molina, J. D. Ullman, and J. Widom, "Database systems: the complete book", 2009.

[59] C. J. Date and H. Darwen, *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. Addison-Wesley, 1989.

[60] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, *Fundamentals of data warehouses*. Springer Science & Business Media, 2013.

[61] A. A. Hancock, E. N. Bush, D. Stanisic, J. J. Kyncl, and C. T. Lin, "Data normalization before statistical analysis: keeping the horse before the cart", *Trends in pharmacological sciences*, vol. 9, no. 1, pp. 29–32, 1988.

[62] P. F. Velleman, "Definition and comparison of robust nonlinear data smoothing algorithms", *Journal of the American Statistical Association*, vol. 75, no. 371, pp. 609–615, 1980.

[63] S. García, J. Luengo, and F. Herrera, "Dealing with Missing Values", in *Data Preprocessing in Data Mining*, Cham: Springer International Publishing, 2015, pp. 59–105, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_4. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4_4.

[64] J. Scheffer, "Dealing with missing data", 2002.

[65] S. García, J. Luengo, and F. Herrera, "Dealing with Noisy Data", in *Data Preprocessing in Data Mining*, Cham: Springer International Publishing, 2015, pp. 107–145, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_5. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4_5.

[66] S. García, J. Luengo, and F. Herrera, "Data Reduction", in *Data Preprocessing in Data Mining*, Cham: Springer International Publishing, 2015, pp. 147–162, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_6. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4_6.

[67] G. H. Dunteman, *Principal components analysis*, 69. Sage, 1989.

[68] S. García, J. Luengo, and F. Herrera, "Feature Selection", in *Data Preprocessing in Data Mining*, Cham: Springer International Publishing, 2015, pp. 163–193, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_7. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4_7.

[69] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[70] S. García, J. Luengo, and F. Herrera, "Instance Selection", in *Data Preprocessing in Data Mining*, Cham: Springer International Publishing, 2015, pp. 195–243, ISBN: 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4_8. [Online]. Available: https://doi.org/10.1007/978-3-319-10247-4_8.

[71] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: towards automating data science endeavors", in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, IEEE, 2015, pp. 1–10.

[72] A. Famili, W.-M. Shen, R. Weber, and E. Simoudis, "Data preprocessing and intelligent data analysis", *Intelligent data analysis*, vol. 1, no. 1, pp. 3–23, 1997.

[73] A. Berson and S. J. Smith, *Data warehousing, data mining, and OLAP*. McGraw-Hill, Inc., 1997.

[74] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules",

[75] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC press, 2013.

[76] A. K. Jain, "Data clustering: 50 years beyond k-means", *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[77] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering", *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[78] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms", *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983.

[79] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[80] M. Ilango and V. Mohan, "A survey of grid based clustering algorithms", *International Journal of Engineering Science and Technology*, vol. 2, no. 8, pp. 3441–3446, 2010.

[81] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: a review of classification techniques", *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[82] A. Shmilovici, "Support vector machines", in *Data mining and knowledge discovery handbook*, Springer, 2009, pp. 231–247.

[83] R. Hecht-Nielsen, "Theory of the backpropagation neural network", in *Neural networks for perception*, Elsevier, 1992, pp. 65–93.

[84] O. Kramer, "K-nearest neighbors", in *Dimensionality reduction with unsupervised nearest neighbors*, Springer, 2013, pp. 13–23.

[85] J. R. Quinlan, "Induction of decision trees", *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[86] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[87] J. Kittler, *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21-23, 2000 Proceedings*. Springer Science & Business Media, 2000, vol. 1857.

[88] F. Stulp and O. Sigaud, "Many regression algorithms, one unified model: a review", *Neural Networks*, vol. 69, pp. 60–79, 2015.

[89] D. F. Specht, "A general regression neural network", *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[90] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines", in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 9*, MIT Press, 1997, pp. 155–161.

[91] J. Pearl, "Fusion, propagation, and structuring in belief networks", *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[92] R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media, 2006.

[93] J. Gebhardt and R. Kruse, "Learning possibilistic networks from data", in *Learning from Data*, Springer, 1996, pp. 143–153.

[94] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data", *Machine learning*, vol. 9, no. 4, pp. 309–347, 1992.

[95] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[96] J. Han, J. Pei, and M. Kamber, "Outlier detection", in *Data mining: concepts and techniques*. Elsevier, 2011, pp. 543–584.

[97] U. M. Fayyad, A. Wierse, and G. G. Grinstein, *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann, 2002.

[98] R. J. Hilderman and H. J. Hamilton, *Knowledge discovery and interestingness measures: A survey*. Citeseer, 1999.

[99] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation", 2011.

[100] R. J. Hilderman and H. J. Hamilton, "Heuristic measures of interestingness", in *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 1999, pp. 232–241.

[101] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: requirements and a survey of the state of the art", *Web Semantics: science, services and agents on the World Wide Web*, vol. 4, no. 1, pp. 14–28, 2006.

[102] N. Guarino, *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*. IOS press, 1998, vol. 46.

[103] J. Lin and B. Katz, "Question answering from the web using knowledge annotation and knowledge mining techniques", in *Proceedings of the twelfth international conference on Information and knowledge management*, ACM, 2003, pp. 116–123.

[104] L. A. Kurgan and P. Musilek, "A survey of knowledge discovery and data mining process models", *The Knowledge Engineering Review*, vol. 21, no. 1, pp. 1–24, 2006.

[105] H. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, "Big data and its technical challenges", *Communications of the ACM*, vol. 57, no. 7, pp. 86–94, 2014.

[106] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-Khah, and P. Siano, "Iot-based smart cities: a survey", in *Environment and Electrical Engineering (EEEIC), 2016 IEEE 16th International Conference on*, IEEE, 2016, pp. 1–6.

[107] G. Ding and B. Bhargava, "Peer-to-peer file-sharing over mobile ad hoc networks", in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, IEEE, 2004, pp. 104–108.

[108] V. Kumar, *Mobile database systems*. John Wiley & Sons, 2006.

[109] E. Pitoura and B. Bhargava, "A framework for providing consistent and recoverable agent-based access to heterogeneous mobile databases", *ACM Sigmod Record*, vol. 24, no. 3, pp. 44–49, 1995.

[110] S. Nishio, M. Tsukamoto, *et al.*, "Data management issues in mobile and peer-to-peer environments", *Data & Knowledge Engineering*, vol. 41, no. 2-3, pp. 183–204, 2002.

[111] G. Bernard, J. Ben-Othman, L. Bouganim, G. Canals, S. Chabridon, B. Defude, J. Ferrié, S. Gançarski, R. Guerraoui, P. Molli, *et al.*, "Mobile databases: a selection of open issues and research directions", *ACM Sigmod Record*, vol. 33, no. 2, pp. 78–83, 2004.

[112] C. Ray, *Distributed database systems*. Pearson Education India, 2009.

[113] M. T. Özsu and P. Valduriez, *Principles of distributed database systems*. Springer Science & Business Media, 2011.

[114] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", *SIGACT News*, vol. 33, no. 2, pp. 51–59, Jun. 2002, ISSN: 0163-5700. DOI: 10.1145/564585.564601. [Online]. Available: http://doi.acm.org/10.1145/564585.564601.

[115] E. Brewer, "Cap twelve years later: how the" rules" have changed", *Computer*, vol. 45, no. 2, pp. 23–29, 2012.

[116] R. P. Padhy, M. R. Patra, and S. C. Satapathy, "Rdbms to nosql: reviewing some next-generation non-relational databases", *International Journal of Advanced Engineering Science and Technologies*, vol. 11, no. 1, pp. 15–30, 2011.

[117] C. Strauch, U.-L. S. Sites, and W. Kriha, "Nosql databases", *Lecture Notes, Stuttgart Media University*, vol. 20, 2011.

[118] W. Golab, M. R. Rahman, A. AuYoung, K. Keeton, and X. S. Li, "Eventually consistent: not what you were expecting?", *Queue*, vol. 12, no. 1, p. 30, 2014.

[119] N. Tang, "Big data cleaning", in *Asia-Pacific Web Conference*, Springer, 2014, pp. 13–24.

[120] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, "Crowdcleaner: data cleaning for multi-version data on the web via crowdsourcing", in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, IEEE, 2014, pp. 1182–1185.

[121] T. Gueta and Y. Carmel, "Quantifying the value of user-level data cleaning for big data: a case study using mammal distribution models", *Ecological informatics*, vol. 34, pp. 139–145, 2016.

[122] G. Cormode and N. Duffield, "Sampling for big data: a tutorial", in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 1975–1975.

[123] A. Satyanarayana, "Intelligent sampling for big data using bootstrap sampling and chebyshev inequality", in *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, IEEE, 2014, pp. 1–6.

[124] H. Zou, Y. Yu, W. Tang, and H. M. Chen, "Improving i/o performance with adaptive data compression for big data applications", in *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, IEEE, 2014, pp. 1228–1237.

[125] O. Dawelbeit and R. McCrindle, "A novel cloud based elastic framework for big data preprocessing", in *Computer Science and Electronic Engineering Conference (CEEC), 2014 6th*, IEEE, 2014, pp. 23–28.

[126] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data", *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[127] G. S. Almasi and A. Gottlieb, "Highly parallel computing", 1988.

[128] D. Taniar and J. W. Rahayu, "Parallel data mining", in *Data mining: A heuristic approach*, IGI Global, 2002, pp. 261–289.

[129] M. J. Zaki, *Large-scale parallel data mining*, 1759. Springer Science & Business Media, 2000.

[130] G. Tsoumakas and I. Vlahavas, "Distributed data mining", in *Database Technologies: Concepts, Methodologies, Tools, and Applications*, IGI Global, 2009, pp. 157–164.

[131] V. Fiolet and B. Toursel, "Distributed data mining", *Scalable Computing: Practice and Experience*, vol. 6, no. 1, 2005.

[132] B.-H. Park and H. Kargupta, "Distributed data mining: algorithms, systems, and applications", 2002.

[133] B. K. Kiran and A. V. Babu, "A comparative study of issues in big data clustering algorithm with constraint based genetic algorithm for associative clustering", *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 8, pp. 5423–5432, 2014.

[134] J. D. Miller, *Big Data Visualization*. Packt Publishing Ltd, 2017.

[135] K.-L. Ma and S. Parker, "Massively parallel software rendering for visualizing large-scale data sets", *IEEE Computer Graphics and Applications*, vol. 21, no. 4, pp. 72–83, 2001.

[136] D. C. Brabham, "Crowdsourcing as a model for problem solving: an introduction and cases", *Convergence*, vol. 14, no. 1, pp. 75–90, 2008.

[137] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk", in *Proceedings of the SIGCHI conference on human factors in computing systems*, ACM, 2008, pp. 453–456.

[138] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing", *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[139] M. Chen, S. Mao, and Y. Liu, "Big data: a survey", *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, Apr. 2014, ISSN: 1572-8153. DOI: 10.1007/s11036-013-0489-0. [Online]. Available: https://doi.org/10.1007/s11036-013-0489-0.

[140] S. Ghemawat, H. Gobioff, and S.-T. Leung, *The Google file system*, 5. ACM, 2003, vol. 37.

[141] M. K. McKusick and S. Quinlan, "Gfs: evolution on fast-forward", *Queue*, vol. 7, no. 7, p. 10, 2009.

[142] D. Borthakur *et al.*, "Hdfs architecture guide", *Hadoop Apache Project*, vol. 53, pp. 1–13, 2008.

[143] H. Li, A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica, "Tachyon: reliable, memory speed storage for cluster computing frameworks", in *Proceedings of the ACM Symposium on Cloud Computing*, ACM, 2014, pp. 1–15.

[144] S. Amazon, *Amazon simple storage service (amazon s3)*, 2012.

[145] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, "Scope: easy and efficient parallel processing of massive data sets", *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1265–1276, 2008.

[146] D. Beaver, S. Kumar, H. C. Li, J. Sobel, P. Vajgel, *et al.*, "Finding a needle in haystack: facebook's photo storage.", in *OSDI*, vol. 10, 2010, pp. 1–8.

[147] H. Zhang, "Research and implementation of distributed file storage system based on tfs", 2016.

[148] C. Plessl and M. Platzner, "Virtualization of hardware-introduction and survey.", in *ERSA*, Citeseer, 2004, pp. 63–69.

[149] Y. Xing and Y. Zhan, "Virtualization and cloud computing", in *Future Wireless Networks and Information Systems*, Springer, 2012, pp. 305–312.

[150] Y. Li, W. Li, and C. Jiang, "A survey of virtual machine system: current technology and future trends", in *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, IEEE, 2010, pp. 332–336.

[151] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics", *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, 2014.

[152] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics", *Journal of Big Data*, vol. 2, no. 1, p. 8, 2015.

[153] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, *Peer-to-peer computing*, 2002.

[154] A. W.-S. Loo, *Peer-To-Peer Computing*. Springer, 2007.

[155] W. D. Gropp, W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface*. MIT press, 1999, vol. 1.

[156] J. Verbeke, N. Nadgir, G. Ruetsch, and I. Sharapov, "Framework for peer-to-peer distributed computing in a heterogeneous, decentralized environment", in *International Workshop on Grid Computing*, Springer, 2002, pp. 1–12.

[157] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[158] A. Hadoop, *Hadoop*, 2009. [Online]. Available: `http://hadoop.%20apache.%20org` (visited on 12/06/2017).

[159] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis", in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, ACM, 2009, pp. 165–178.

[160] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: efficient iterative data processing on large clusters", *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 285–296, 2010.

[161] J. Ekanayake, S. Pallickara, and G. Fox, "Mapreduce for data intensive scientific analyses", in *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, IEEE, 2008, pp. 277–284.

[162] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: a runtime for iterative mapreduce", in *Proceedings of the 19th ACM international symposium on high performance distributed computing*, ACM, 2010, pp. 810–818.

[163] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "Imapreduce: a distributed computing framework for iterative computation", *Journal of Grid Computing*, vol. 10, no. 1, pp. 47–68, 2012.

[164] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark: a unified engine for big data processing", *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[165] Y. M. Essa, G. Attiya, and A. El-Sayed, "Mobile agent based new framework for improving big data analysis", in *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, IEEE, 2013, pp. 381–386.

[166] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks", in *ACM SIGOPS operating systems review*, ACM, vol. 41, 2007, pp. 59–72.

[167] H. Li, G. Fox, and J. Qiu, "Performance model for parallel matrix multiplication with dryad: dataflow graph runtime", in *Cloud and Green Computing (CGC), 2012 Second International Conference on*, IEEE, 2012, pp. 675–683.

[168] A. Storm, "Storm, distributed and fault-tolerant realtime computation", *Google Scholar*, 2013.

[169] W. Yang, X. Liu, L. Zhang, and L. T. Yang, "Big data real-time processing based on storm", in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, IEEE, 2013, pp. 1784–1787.

[170] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: distributed stream computing platform", in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, IEEE, 2010, pp. 170–177.

[171] J. Chauhan, S. A. Chowdhury, and D. J. Makaroff, "Performance evaluation of yahoo! s4: a first look.", in *3PGCIC*, 2012, pp. 58–65.

[172] R. Buyya *et al.*, "High performance cluster computing: architectures and systems (volume 1)", *Prentice Hall, Upper SaddleRiver, NJ, USA*, vol. 1, p. 999, 1999.

[173] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[174] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: maximizing on-chip parallelism", in *ACM SIGARCH Computer Architecture News*, ACM, vol. 23, 1995, pp. 392–403.

[175] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing", *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[176] C. Nvidia, "Nvidia cuda c programming guide", *Nvidia Corporation*, vol. 120, no. 18, p. 8, 2011.

[177] W. Fang, K. K. Lau, M. Lu, X. Xiao, C. K. Lam, P. Y. Yang, B. He, Q. Luo, P. V. Sander, and K. Yang, "Parallel data mining on graphics processors", *Hong Kong Univ. Sci. and Technology, Hong Kong, China, Tech. Rep. HKUST-CS08-07*, 2008.

[178] R. J. Francis, J. Rose, and K. Chung, "Chortle: a technology mapping program for lookup table-based field programmable gate arrays", in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, ACM, 1991, pp. 613–619.

[179] D. Thomas and P. Moorby, *The Verilog® Hardware Description Language*. Springer Science & Business Media, 2008.

[180] R. Griesemer, "Parallelism by design: data analysis with sawzall", in *Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization*, ACM, 2008, pp. 3–3.

[181] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing", in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM, 2008, pp. 1099–1110.

[182] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive-a petabyte scale data warehouse using hadoop", in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, IEEE, 2010, pp. 996–1005.

[183] Y. Y. M. I. D. Fetterly, M. Budiu, Ú. Erlingsson, and P. K. G. J. Currey, "Dryadlinq: a system for general-purpose distributed data-parallel computing using a high-level language", *Proc. LSDS-IR*, p. 8, 2009.

[184] M. Franklin, "Making sense of big data with the berkeley data analytics stack", in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ACM, 2015, pp. 1–2.

[185] C. Ma, H. H. Zhang, and X. Wang, "Machine learning for big data analytics in plants", *Trends in plant science*, vol. 19, no. 12, pp. 798–808, 2014.

[186] R. C. Team *et al.*, "R: a language and environment for statistical computing", 2014.

[187] S. R. Garner *et al.*, "Weka: the waikato environment for knowledge analysis", in *Proceedings of the New Zealand computer science research students conference*, 1995, pp. 57–64.

[188] R. Bouman and J. Van Dongen, *Pentaho solutions: business intelligence and data warehousing with Pentaho and MySQL*. Wiley Publishing, 2009.

[189] M. Hofmann and R. Klinkenberg, *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2013.

[190] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "Knime-the konstanz information miner: version 2.0 and beyond", *AcM SIGKDD explorations Newsletter*, vol. 11, no. 1, pp. 26–31, 2009.

[191] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: machine learning in python", *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[192] D. Albanese, R. Visintainer, S. Merler, S. Riccadonna, G. Jurman, and C. Furlanello, "Mlpy: machine learning python", *arXiv preprint arXiv:1202.6548*, 2012.

[193] R. R. Curtin, J. R. Cline, N. P. Slagle, W. B. March, P. Ram, N. A. Mehta, and A. G. Gray, "Mlpack: a scalable c++ machine learning library", *Journal of Machine Learning Research*, vol. 14, no. Mar, pp. 801–805, 2013.

[194] S. Sonnenburg, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, V. Franc, *et al.*, "The shogun machine learning toolbox", *Journal of Machine Learning Research*, vol. 11, no. Jun, pp. 1799–1802, 2010.

[195] A. Mahout, *Scalable machine learning and data mining*, 2012.

[196] Z. Prekopcsak, G. Makrai, T. Henk, and C. Gaspar-Papanek, "Radoop: analyzing big data with rapidminer and hadoop", in *Proceedings of the 2nd RapidMiner community meeting and conference (RCOMM 2011)*, 2011, pp. 1–12.

[197] S. MLlib, "Apache spark's scalable machine learning library", [Online]. Available: `http://spark.%20apache.%20org/mllib` (visited on 12/02/2017).

[198] S. Pochampalli, *Jaspersoft bi suite tutorials*, 2016.

[199] D. TÂRNĂVEANU, "Pentaho business analytics: a business intelligence open source alternative", *Database Systems Journal*, vol. 3, no. 3, p. 23, 2012.

[200] D. Samuels, *Skytree: machine learning meets big data, february 2012*.

[201] A. Rueter and S. Solutions, "Tableau for the enterprise: an overview for it", *Tableau Software*, 2012.

[202] J. Bowen, *Getting Started with Talend Open Studio for Data Integration*. Packt Publishing Ltd, 2012.

[203] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: a technology tutorial", *IEEE access*, vol. 2, pp. 652–687, 2014.

[204] S. Stream, "Product page (sql stream ramms) as of jun. 13, 2008", *SQL Stream Incorporated (available as of Dec. 20, 2010 at https://web. archive. org/web/2008061332512/www. sglstream. com/products/productoffersRAMMS. htm)*,

[205] T. Samson, *Splunk storm brings log management to the cloud*, 2012.

[206] N. Garg, *Apache Kafka*. Packt Publishing Ltd, 2013.

[207] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees, "The sap hana database–an architecture overview.", *IEEE Data Eng. Bull.*, vol. 35, no. 1, pp. 28–33, 2012.

[208] L. Garber, "Using in-memory analytics to quickly crunch big data", *Computer*, vol. 45, no. 10, pp. 16–18, 2012.

[209] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co., 2015.

[210] Hunter Heaivilin and Will Weider, *Big Data in Science & Problem Solving*, en, 2018. [Online]. Available: https://international-soil-radiocarbon-database.github.io//SOC-Hub//2018/06/27/Big-Data/ (visited on 10/11/2018).

[211] *2018 reform of EU data protection rules*, en, Text, 2018. [Online]. Available: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en (visited on 07/09/2018).

[212] M. Hilbert and P. López, "The World's Technological Capacity to Store, Communicate, and Compute Information", *Science*, vol. 332, no. 6025, pp. 60–65, 2011, ISSN: 0036-8075. DOI: 10.1126/science.1200970. [Online]. Available: http://science.sciencemag.org/content/332/6025/60.

[213] *SIGKDD - KDD Cup*, 2018. [Online]. Available: http://www.kdd.org/kdd-cup (visited on 02/10/2018).

[214] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms", *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[215] *IJCAI-18 Alimama International Advertising Algorithm Competition*, 2018. [Online]. Available: `https://tianchi.aliyun.com/markets/tianchi/ijcai2018en?spm=a2c41.games_official.0.0` (visited on 02/10/2018).

[216] *ECML PKDD | European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2018. [Online]. Available: `http://www.ecmlpkdd.org/` (visited on 02/10/2018).

[217] *Texata - Big Data World Championships*, en-US, 2018. [Online]. Available: `http://www.texata.com/` (visited on 02/10/2018).

[218] V. Bhatnagar and S. Srinivasa, *Big Data Analytics: Second International Conference, BDA 2013, Mysore, India, December 16-18, 2013, Proceedings*. Springer, 2013, vol. 8302.

[219] S. Srinivasa and S. Mehta, *Big Data Analytics: Third International Conference, BDA 2014, New Delhi, India, December 20-23, 2014. Proceedings*. Springer, 2015, vol. 8883.

[220] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future", *ACM sIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1–5, 2013.

[221] A. Paprotny and M. Thess, *Realtime Data Mining: Self-learning Techniques for Recommendation Engines*. Springer Science & Business Media, 2013.

[222] T. Palpanas, "Data series management: the road to big sequence analytics", *ACM SIGMOD Record*, vol. 44, no. 2, pp. 47–52, 2015.

[223] B. Tidke and R. Mehta, "A comprehensive review and open challenges of stream big data", in *Soft Computing: Theories and Applications*, Springer, 2018, pp. 89–99.

[224] V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam, T. Bergauer, M. Dragicevic, J. Erö, C. Fabjan, M. Friedl, R. Frühwirth, *et al.*, "Observation of the diphoton decay of the higgs boson and measurement of its properties", *The European Physical Journal C*, vol. 74, no. 10, p. 3076, 2014.

[225] G. Weiss, *Multiagent Systems, A modern Approach to Distributed Artificial Systems*. MIT Press, 1999, ISBN: 0-262-23203-0.

[226] L. Cao, V. Gorodetsky, and P. A. Mitkas, "Agent mining: the synergy of agents and data mining", *IEEE Intelligent Systems*, vol. 24, no. 3, 2009.

[227] F. Bromberg, V. Honavar, and D. Caragea, *Multi-agent data mining*, 2004.

[228] V. Gorodetsky, O. Karsaeyv, and V. Samoilov, "Multi-agent technology for distributed data mining and classification", in *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, IEEE, 2003, pp. 438–441.

[229] L. Cao, A. Gorodetsky, J. Liu, G. Weiss, and S. Y. Philipp, *Agents and data mining interaction*. Springer, 2009.

[230] A. Chemchem and H. Drias, "From data mining to knowledge mining: application to intelligent agents", *Expert Systems with Applications*, vol. 42, no. 3, pp. 1436–1445, 2015.

[231] L. Cao, A. L. Bazzan, A. L. Symeonidis, V. Gorodetsky, G. Weiss, and S. Y. Philip, *Agents and Data Mining Interaction: 7th International Workshop, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*. Springer, 2011, vol. 7103.

[232] L. Cao, Y. Zeng, A. L. Symeonidis, V. Gorodetsky, S. Y. Philip, and M. P. Singh, *Agents and Data Mining Interaction: 8th International Workshop, ADMI 2012, Valencia, Spain, June 4-5, 2012, Revised Selected Papers*. Springer, 2013, vol. 7607.

[233] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Metaheuristics for Hard Optimization*. Springer, 2006.

[234] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1998, ISBN: 978-0-262-63185-3.

[235] J. Lawrence, *Introduction to Neural Networks*. California Scientific, Nevada City, CA 95959., 1993, ISBN: 1-883157-00-5.

[236] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Eds., *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering*, ser. LNCS. Springer, 2004, vol. 2977, ISBN: 3-540-21201-9.

[237] S. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, Eds., *Engineering Self-Organising Systems*, ser. LNAI. Springer, 2006, vol. 3910.

[238] N. Ashish and J.-L. Ambite, *Data Integration in the Life Sciences: 11th International Conference, DILS 2015, Los Angeles, CA, USA, July 9-10, 2015, Proceedings*. Springer, 2015, vol. 9162.

[239] Z. Sehili, L. Kolb, C. Borgs, R. Schnell, and E. Rahm, "Privacy preserving record linkage with ppjoin.", in *BTW*, 2015, pp. 85–104.

[240] G. Vossen, "Big data as the new enabler in business and other intelligence", *Vietnam Journal of Computer Science*, vol. 1, no. 1, pp. 3–14, 2014.

[241] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential", *Health Information Science and Systems*, vol. 2, no. 1, p. 3, 2014.

[242] Z. Khan, A. Anjum, K. Soomro, and M. A. Tahir, "Towards cloud based big data analytics for smart future cities", *Journal of Cloud Computing*, vol. 4, no. 1, pp. 1–11, 2015.

[243] E. Al Nuaimi, H. Al Neyadi, N. Mohamed, and J. Al-Jaroodi, "Applications of big data to smart cities", *Journal of Internet Services and Applications*, vol. 6, no. 1, pp. 1–15, 2015.

[244] N. Verstaevel, "Self-Organization of Robotic Devices Through Demonstrations", PhD thesis, Université de Toulouse, Toulouse, France, Jun. 2016. [Online]. Available: `http://thesesups.ups-tlse.fr/3249/1/2016TOU30060.pdf`.

[245] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-organising software: from natural to artificial adaptation", 2011.

[246] J. P. Georgé, B. Edmonds, and P. Glize, "Making self-organizing adaptive multi-agent systems work", in *Methodologies and Software Engineering for Agent Systems (Chapter 16)*, F. Bergenti, M. P. Gleizes, and F. Zambonelli, Eds. Kluwer Publishing, 2004, ch. 16, pp. 321–340.

[247] M.-P. Gleizes, V. Camps, and P. Glize, "A Theory of Emergent Computation Based on Cooperative Self-Oganization for Adaptive Artificial Systems", in 4^{th} *European Congress of Systems Science*, 1999.

[248] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos, "History and definitions", in *Self-organising Software*, Springer, 2011, pp. 33–74.

[249] P. A. Corning, "The re-emergence of emergence, and the causal role of synergy in emergent evolution", en, *Synthese*, vol. 185, no. 2, pp. 295–317, Mar. 2012, WOS:000303530600009, ISSN: 0039-7857, 1573-0964. DOI: 10.1007/s11229-010-9726-2. [Online]. Available: https://link.springer.com/article/10.1007/s11229-010-9726-2 (visited on 04/05/2017).

[250] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.

[251] G. Picard and M.-P. Gleizes, "Cooperative Self-Organization to Design Robust and Adaptive Collectives", in *2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO'05), Barcelona, Spain, Volume I*, INSTICC Press, 2005.

[252] B. Hayes-Roth, "A blackboard architecture for control", *Artificial intelligence*, vol. 26, no. 3, pp. 251–321, 1985.

[253] B. E. Cooper, "Correlation and Function Fitting", en, in *Statistics for Experimentalists*, Elsevier, May 2014, pp. 206–212, ISBN: 978-1-4832-8052-3.

[254] D. D. Nolte, "The tangled tale of phase space", en, *Physics Today*, vol. 63, no. 4, pp. 33–38, Apr. 2010, ISSN: 0031-9228, 1945-0699. DOI: 10.1063/1.3397041. (visited on 04/04/2017).

[255] T. Finch, "Incremental calculation of weighted mean and variance", *University of Cambridge*, vol. 4, pp. 11–5, 2009.

[256] SARL, *Basic Object-Oriented Programming Support*, Sep. 2018. [Online]. Available: http://www.sarl.io/docs/official/reference/OOP.html%5C#1-comparison-between-sarl-and-other-languages (visited on 10/16/2017).

[257] S. Rodriguez, N. Gaud, and S. Galland, "SARL: a general-purpose agent-oriented programming language", in *the 2014 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Warsaw, Poland: IEEE Computer Society Press, 2014.

[258] J. Boes, J. Nigon, N. Verstaevel, M.-P. Gleizes, and F. Migeon, "The self-adaptive context learning pattern: overview and proposal (regular paper)", anglais, in *International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), Larnaca, Cyprus,*, ser. LNAI, Springer, 2015, pp. 91–104.

[259] Y. Motie, E. Belghache, A. Nketsa, and J.-P. Georgé, "Interoperability based dynamic data mediation using adaptive multi-agent systems for co-simulation", in *2018 International Conference on High Performance Computing & Simulation (HPCS)*, IEEE, 2018, pp. 235–241.

[260] R. Cazabet and F. Amblard, "Simulate to detect: a multi-agent system for community detection", in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*, IEEE Computer Society, 2011, pp. 402–408.

[261] M.-P. Gleizes, J. Boes, B. Lartigue, and F. Thiébolt, "Neocampus: a demonstrator of connected, innovative, intelligent and sustainable campus", in *International Conference on Intelligent Interactive Multimedia Systems and Services*, Springer, 2017, pp. 482–491.

[262] *The neOCampus projey of the University of Toulouse III - Paul Sabatier*, 2017. [Online]. Available: https://neocampus.univ-tlse3.fr (visited on 07/10/2017).

[263] *UCI Machine Learning Repository: Data Sets*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.html?format=%5C&task=%5C&att=%5C&area=%5C&numAtt=%5C&numIns=%5C&type=ts%5C&sort=nameUp%5C&view=list (visited on 08/01/2017).

[264] *UCI Machine Learning Repository: Ozone Level Detection Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection (visited on 08/01/2017).

[265] *UCI Machine Learning Repository: ElectricityLoadDiagrams20112014 Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 (visited on 08/01/2017).

[266] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[267] Walber, *File:Precisionrecall.svg - Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/File:Precisionrecall.svg (visited on 06/07/2018).

[268] IRIT, *OSIRIM Platform*, 2018. [Online]. Available: http://osirim.irit.fr/site/en/articles/welcome (visited on 03/07/2018).

[269] CALMIP, *CALMIP*, 2018. [Online]. Available: https://www.calmip.univ-toulouse.fr/ (visited on 07/02/2018).

[270] E. O. Brigham and E. O. Brigham, *The fast Fourier transform and its applications*. prentice Hall Englewood Cliffs, NJ, 1988, vol. 448.

[271] Phonical, *Fast Fourier Transform [cc by-sa 4.0]*, Nov. 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:FFT-Time-Frequency-View.png (visited on 11/30/2017).

[272] P. J. Rousseeuw and K. V. Driessen, "A fast algorithm for the minimum covariance determinant estimator", *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[273] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Robust covariance estimation and Mahalanobis distances relevance - scikit-learn 0.21.0 documentation*, 2011. [Online]. Available: https://scikit-learn.org/stable/auto_examples/covariance/plot_mahalanobis_distances.html (visited on 09/25/2017).

List of Tables

List of Figures

Glossary

Cardinality: the number of elements in a set or other grouping. 94, 96, 97, 102, 106, 107, 127, 130, 132, 135, 136, 140, 143

Counting: the occurrence number of an element in a set or other grouping. 88

AMAS Adaptive Multi-agent System (cf. 6). 75, 79, 80, 81, 83, 84, 88, 90, 109, 214

AMAS4BigData Adaptive Multi-Agent Systems for Dynamic Big Data Analytics (cf. 6.4). 69, 79, 84, 85, 87, 88, 90, 94, 95, 103, 108, 109, 115, 128, 135, 136, 137, 142, 143, 144, 147, 149, 153, 155, 156, 159, 161, 162, 163, 164, 171, 172, 173, 174, 175, 176, 177, 178, 214

DC Dynamics Correlation (cf. 7.1.3). 127, 130, 132, 133, 136, 139, 140

DREAM Dynamic data Relation Extraction using Adaptive Multi-agent systems (cf. 7). 115, 116, 128, 129, 130, 132, 135, 136, 137, 139, 140, 141, 142, 143, 144, 147, 149, 150, 157, 161, 162, 177, 178, 185, 215, 216

IoT Internet of Things (cf. 3.1.1). 34, 36, 64, 69, 84, 142, 149, 153, 154, 162, 164, 172, 175, 215

KDD Knowledge Discovery from Data (cf. 2). 10, 12, 19, 20, 21, 31, 54, 62, 73, 76, 213

LPSS Local Phase Space Similarity (cf. 7.1.2.3). 126, 127, 130, 215

MAS Multi-agent System (cf. 6). 73, 80, 81, 82, 140, 142

NCS Non Cooperative Situations (cf. 5). 82, 83, 95, 103, 105, 106, 214

NoSQL Not only Structured Query Language (cf. 3.1.1.3). 35, 36, 40

PS Phase Space (cf. 7.1.2.1). 119

PSS Phase Space Similarity (cf. 7.1.2.2). 124, 125, 126, 127, 215

SI Situation of Interest (cf. 7.1.2.3). 126, 127, 130, 132, 133, 134, 140