Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Unleashing Constraint Optimisation Problem solving in Big Data environments

Álvaro Valencia-Parra[a], Ángel Jesús Varela-Vaca[a,*], Luisa Parody[b], María Teresa Gómez-López[a,1]

[a] Dpto. Lenguajes y Sistemas Informáticos, Universidad de Sevilla, IDEA Research group, Spain
[b] Dpto. Métodos Cuantitativos, Universidad Loyola Andalucía, IDEA Research group, Spain

ARTICLE INFO

Article history:
Received 25 November 2019
Received in revised form 2 June 2020
Accepted 18 June 2020
Available online 24 June 2020

Keywords:
Big Data
Optimisation problem
Constraint programming
Distributed data
Heterogeneous data format

ABSTRACT

The application of the optimisation problems in the daily decisions of companies is able to be used for finding the best management according to the necessities of the organisations. However, optimisation problems imply a high computational complexity, increased by the current necessity to include a massive quantity of data (Big Data), for the creation of optimisation problems to customise products and services for their clients. The irruption of Big Data technologies can be a challenge but also an important mechanism to tackle the computational difficulties of optimisation problems, and the possibility to distribute the problem performance. In this paper, we propose a solution that lets the query of a data set supported by Big Data technologies that imply the resolution of Constraint Optimisation Problem (COP). This proposal enables to: (1) model COPs whose input data are obtained from distributed and heterogeneous data; (2) facilitate the integration of different data sources to create the COPs; and, (3) solve the optimisation problems in a distributed way, to improve the performance. It is done by means of a framework and supported by a tool capable of modelling, solving and querying the results of optimisation problems. The tool integrates the Big Data technologies and commercial solvers of constraint programming. The suitability of the proposal and the development have been evaluated with real data sets whose computational study and results are included and discussed.

1. Introduction

The use of optimisation problems let organisations manage the resources, time, and cost of their processes. However, the necessity to create customised products according to the profiles of the clients implies the creation of thousands of optimisation problems. Moreover, the incorporation of more interesting data, frequently provided by multiple systems and services [33], will make companies more competitive. Moreover, the integration of more complex data implies the resolution of optimisation problems that could suppose a computationally complex task, further an extra effort to integrate heterogeneous data format provided by different sources. COPs are frequently used to model and solve optimisation problems since they include among their advantages: the ability to model problems declaratively, regardless of how it will be solved; their applicability to many real-life examples of industry, so its versatility and power are empirically demonstrated; and the existence of a significant amount of commercial tools that solve the constraint satisfaction problems locally. The resolution time of the problem can be compromised for both the complexity of the constraints and the size of the data set. Thereby, in the current scenarios, the information involved in a COP is not always centralised. The exponential growth of the data produced and stored and the distribution of the information have promoted the use of Big Data paradigm [25,31], producing difficulties that have not been adapted to COPs [41]. This new situation can produce that, the data and constraints that model their relations are distributed among different subsystems (also known as nodes) that constitute the global model. These models can be computationally highly complex since they can have distributed data or constraints. This interruption of Big Data into COPs can be seen as a new challenge, but also as a new opportunity to solve distributed problems and data.

Before the Big Data burst into the scene, Distributed Constraint Optimisation Problems (DCOPs) [21,11] were defined to solve COPs where neither their constraints nor data were in a single system.

* Corresponding author.
E-mail addresses: avalencia@us.es (Á. Valencia-Parra), ajvarela@us.es (Á.J. Varela-Vaca), mlparody@uloyola.es (L. Parody), maytegomez@us.es (M.T. Gómez-López).
1 http://www.idea.us.es/

https://doi.org/10.1016/j.jocs.2020.101180
1877-7503/© 2020 Elsevier B.V. All rights reserved.

DCOP paradigm proposes a set of algorithms to solve a set of COPs that share variables. The resolution of each of them must be aware of the rest of the constraints that restrict the values of the variables. For this reason, DCOPs are focused on the communication between the distributed nodes that must synchronise the share variables to obtain a correct result of every constraint, variable and in each node. However, DCOP proposals do not provide a mechanism for a better distribution of the DCOPs for the optimal resolution of them. In this context, our proposal is based on the creation and distribution of isolated COPs in a Big Data infrastructure, whose partial resolution provides information to solve a greater problem utilising the MapReduce paradigm [10]. Furthermore, none of the algorithms proposed to solve DCOP have been presented as a practical and appropriate solution to actual environments, even less when the data used is characterised by its high variability, velocity, and volume (i.e., Big Data). Although the existence of the problem was formulated in our previous works [35,36], to the best of our knowledge, there are no solutions that adequate the COPs to data supported by Big Data infrastructure, dealing with large amounts of data, using the distributed node to parallelise the computation, getting results quickly, and managing a variety of data efficiently.

As aforementioned, we identified the necessity of the COP resolution within different Big Data scenarios in [35,36]. In those previous work, we focused on describing the scenarios and their characteristics, then a solution was prototyped. However, the previous work lacks formalisation, implementation and evaluation with real data was done.

In order to fulfil those gaps, we propose a solution which enables to define, execute and solve Constraint Optimisation Problems with Distributed Data in Big Data environment. The main contributions are four main:

- **Formalisation of Constraint Optimisation Problems with Distributed (Big) Data**: An optimisation problem must be modelled describing the constraints that relate the data and the objective function to be optimised. When the data is distributed and the resolution of the problems depends on this data, the description of the model must include the distributed data than can have heterogeneous formats and come from Big Data environments. We formalise COP models that include these aspects.
- **Integration and transformation of heterogeneous data formats**: The heterogeneity of the data formats makes necessary data preparation by means of a transformation into a unified format to apply the latter analysis. This task is especially relevant in our proposal since the mapping between input data to the optimisation problem must be defined, at the same time that the output obtained must be recovered.
- **Enable the optimal evaluation of queries over data provided by the resolution of Optimisation Problems**: The resolution of various COPs can provide useful information, for example, to ascertain the most suitable products for each provider. However, to select the most appropriate for a specific organisation, these partial solutions obtained from the evaluation of the COPs must be queried. Our proposal is focused on the analysis of the queries over the COP data output to avoid, when it is possible, the resolution of the COPs, minimising the evaluation time.
- **Provide an integral framework to support the process**: To cover the previous proposals, it is necessary a leap of advances in technologies that support the integration of the Big Data infrastructures with constraint optimisation solvers. This new technology is proposed as a new component in the Big Data ecosystem to enable the management of a great amount of data and the creation and distribution of several COPs, whose output data can be later queried.

  Unfortunately, there are no technological solutions that enable the application of optimisation problems with Big Data characteristics in real scenarios. This lack of tools means that these challenges need an extra effort to be solved.

In order to tackle our proposal, FAst BIg cOstraint LAboratory (FABIOLA) framework is proposed. FABIOLA is a new Hadoop-based component [3] in the Big Data ecosystem. FABIOLA enables the modelling of optimisation problems whose heterogeneous format of the data, the amount of data to deal with, and the required velocity in the resolution of the problems forces a Big Data solution. The FABIOLA framework also provides the necessary techniques to solve COPs in a distributed way, either to obtain a result by optimising time and resources or because the nature of the problem is distributed.

FABIOLA prototype was presented in previous works [35,36], thereby, this paper can be seen as an extension of the previous ones. However, certain challenges tackled here were not tackled in that previous works: how to transform the data for its integration with other distributed data; how to optimise the data obtained from the optimisation by means of a query language that reduces the COPs evaluation, and; how an integral solution could support the whole process providing an implemented tool used for the community as a new component of the Big Data ecosystem, and; the empirical evaluation of the approach with real data.

The remainder of the paper is organised as follows: Section 2 formalises the modelling of the optimisation problems in Big Data environments. Section 3 details the different phases of the proposal that are illustrated with a running example to make the problem understandable. Section 4 analyses various operators for processing the results obtained from the resolution of the optimisation problems. Section 5 details the proposed architecture and the methodology necessary to support FABIOLA framework and the developed infrastructure. Section 6 presents the formalisation applied to a real case with a high volume of data. This section also shows a computational, statistical, and analytical study of our proposal. Section 7 includes the main related work. Finally, conclusions are drawn and future work is proposed in Section 8.

2. Overview of the proposal: Optimisation Problems within Big Data scenario

The creation of a *COPDD* is a complex task, especially when the quantity of data involved, and the number of COPs is very high. This is why this creation must be recommendable automated. For this reason, our proposal integrates a set of components that facilitate both the creation and the distribution of the COPs for their resolution.

Definition 1. A COPDD is defined by the tuple $\langle \text{DS, COP, DM, DMap}_{\text{DS}\rightarrow\text{DM}}, \text{DMap}_{\text{DM}\rightarrow\text{COP}} \rangle \rightarrow \text{DS}^{output}$, where:

- DS is the Data Set used as input data to find out the optimal solution of the problem.
- COP is the Constraint Optimisation Problem.
- DM is the Data Model that describes the relationship between the attributes of the DS and the COP.
- $\text{DMap}_{\text{DS}\rightarrow\text{DM}}$ and $\text{DMap}_{\text{DM}\rightarrow\text{COP}}$ are the Data Mapping (DMap) that represents the relation between the attributes of the DS and the attributes of the DM, and the attributes of the DM and the variables used in the COP, respectively.
- DS^{output} is a data set obtained from the resolution of an optimisation problem according to the attributes defined in the DM and their corresponding values.

Each element of the tuple of the COPDD plays a role to create and solve the COPs in an efficient way, as shown in Fig. 1. They are combined to create a framework that integrates the four phases of the

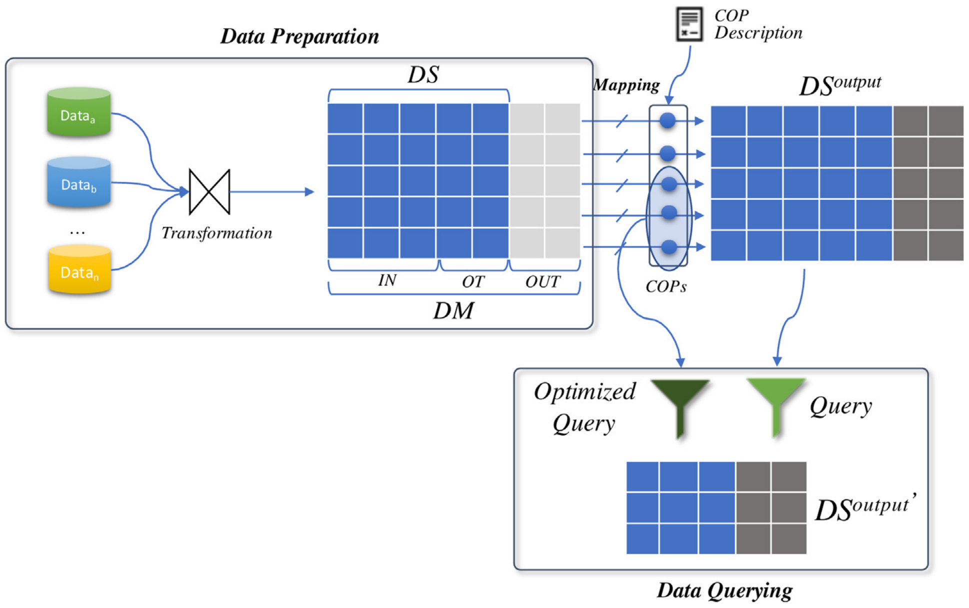


Fig. 1. Overview of the approach.

COPDD creation: (1) data set preparation; (2) COP description; (3) mapping among the data sets and the COP, and; (4) data querying optimising the performance. Some definitions have been included to clarify the phases of the proposal, each of them is detailed in the following sections.

2.1. Running example

To understand the proposal, a simple running example is used, although a real and more complex case study to compute the evaluation is given in Section 6. The running example is about a planning problem [9], that describes a company that manufactures two types of products: a standard product A, and a more sophisticated product B. Management charges a certain price for each unit of product A and a price for the unit of product B, whose profits are p_A and p_B respectively. Therefore, the *profit* of the company comes from the multiplication of the price by the number of units (i.e., q_A and q_B for products A and B) of each product. Manufacturing one unit of product A requires h_A hours of labour and rm_A units of raw material. Similarly, for one unit of B, h_B hours of labour and rm_B units of raw material are needed. Besides, the company's policy indicates that the number of units manufactured of product A has to be, at least, the double of units than the produced for product B. At present, a nH of hours of labour and a nRM of units of raw material are available. The problem is focused on to *maximise* the company's total revenue. Although every element of the COPDD will be described in the following sections, a brief approximation to the formalisation with the running example can be done at this point by identifying the Data Set (DS) and data to optimise:

- DS: $price_A$, $price_B$, $numUnits_A$, $numUnits_B$, $hours_A$, $hours_B$, $hoursLabour$, raw_A, raw_B, $unitsRawMaterial$.
- COP is defined to maximise the *profit*.

3. Phases of the proposal

As presented in Fig. 1, four are the phases that our proposal supports to create and solve the COPDDs. Following sections describe each of them.

3.1. Data set preparation

Data preparation is one of the most consuming processes in the Big Data pipeline [22,44]. One of the steps is the data transformation to fit the heterogeneous data formats into a single one for a later application of algorithms over all tuples. This type of problem is also tackled for the COPDD, that needs a DS with homogeneous tuples used as inputs. It implies that each input data ($Data_a$, $Data_b$, ..., $Data_n$) must be transformed into a single one structure (DS), following the data transformation processes as detailed in [46]. Therefore, a heterogeneous data format is a set of data ($Data_a, Data_b, \ldots, Data_n$) formed of a different set of attributes among them.

Definition 2. A DS is a set of tuples formed of a set of attributes, $\{a_1, a_2, \ldots, a_n\}$, where each tuple presents an assignment of values $\{val_1, val_2, \ldots, val_n\}$ to each attribute.

Some of the elements of the second row of Table 1 (cf., *#ID*, *nH*, *nRM*, h_A, rm_A, p_A, h_B, rm_B, p_B) presents the elements of the DS for the running example.

3.2. Constraint Optimisation Problem description

A COP is a Constraint Satisfaction Problem (CSP) where an optimisation function determines the 'best solution' from the set of possibles. They come from Constraint Programming (CP) [39], an Artificial Intelligence (AI) discipline where a large number of problems and other areas of Computer Science can be seen as individual cases of CSP. Examples include scheduling, temporal reasoning, graph problems, and configuration problems. The basis of CP stands on the resolution of a CSP. Thereby, to understand correctly what is a COP, the CSP definition must be introduced before.

Definition 3. A CSP represents a reasoning framework consisting of variables, domains and constraints $\langle V, D, C\rangle$, where V is a set of n variables $\{v_1, v_2, \ldots, v_n\}$ whose values are taken from finite domains $\{D_{v_1}, D_{v_2}, \ldots, D_{v_m}\}$ respectively, and C is a set of constraints on their values. The constraint $c_k(x_{k_1}, \ldots, x_{k_m})$ is a predicate that is defined on the Cartesian product $D_{k_1} \times \ldots \times D_{k_j}$. This predicate is true iff the value assignment of these variables satisfies the constraint c_k.

Table 1
Example of tuples for the running example.

	IN								OUT			OT
#ID	*nH*	*nRM*	h_A	rm_A	p_A	h_B	rm_B	p_B	***profit***	q_A	q_B	Comp.
#1	1500	400	3	4	5	4	6	10	570	58	28	Comp1
#2	900	200	1	2	3	2	3	4	300	100	0	Comp2
#3	1200	–	7	4	14	–	7	30	160	80	4640	Comp2
#4	–	300	2	3	9	4	6	11	900	100	0	Comp3
#5	1500	–	–	1	8	–	–	15	12000	1500	0	Comp4

Table 2
COP for the running example.

Variables &domains:	$price_A$, $price_B$, $profit$: Float;
	$numUnits_A$, $numUnits_B$, $hours_A$, $hours_B$: Integer;
	raw_A, raw_B, $hoursLabour$, $unitsRawMaterial$: Integer;
Constraints:	$\sum_{i=A}^{B}(hours_i * numUnits_i) \leq hoursLabour$
	$\sum_{i=A}^{B}(raw_i * numUnits_i) \leq unitsRawMaterial$
	$numUnits_A - 2 * numUnits_B \geq 0$
	$profit = \sum_{i=A}^{B}(price_i * numUnits_i)$
Optimisation function:	$Maximise(profit)$

The search for solutions for a CSP is based on the instantiation concept. An assignment of a variable, or instantiation, is a pair variable-value (x, a) which represents the assignment of the value a to the variable x. An instantiation of a set of variables is a tuple of ordered pairs, where each sorted pair (x, a) assigns the value a to the variable x. A tuple $((x_1, a_1), \ldots, (x_i, a_i))$ is consistent if it satisfies all the constraints formed by variables of the tuple.

Several possible solutions can be found to satisfy a CSP, but sometimes only the more suitable wants to be obtained, one that optimises the solution. In those cases, a COP must be modelled.

Definition 4. A COP is a CSP in which the solutions optimise (minimise or maximise) an objective function, *f*.

For the example, for maximising the company's total revenue can be formulated as shown in Table 2.

The values of nH, nRM, h_A, rm_A, p_A, h_B, rm_B, p_B depend on the company where the products are built. Table 1 shows some possible scenarios, where the output data (i.e., $profit$, q_A, q_B) is obtained according to each input data (per tuple). Every input data can be instantiated (e.g., tuples # 1 and # 2), and the output indicates the resulting *profit* and how many units of each product produces (cf., q_A and q_B). It is also possible that some input values were unknown (cf., the value '–' in the table). In that case, the COP tries to find values for these variables to optimise the variable *profit* (cf., tuples # 3, # 4 and # 5). In this example, we can observe that there is another attribute, such as *Company* (cf., *Comp.*), that is not part of and does not influence the resolution of the optimisation problem. However, this kind of variables can be helpful to answer later queries, such as, *Which is the manufacturer with the highest profit? Which is the manufacturer that produces more products B?*

If there are several tuples of input data, several are the possible optimal solutions that can be found. Thereby, the created COP can be seen as a meta-COP which is partially instantiated for each tuple of the DS. How to map the attributes of DS with the inputs of the COP is described is explained in the next subsections.

3.3. *The mapping between the DS and COP*

In order to link the DS with the COP, it is necessary to define an independent DM which allow users to define how the data from the DS is used as input of the COPs, and the resultant data obtained by solving the COPs.

Definition 5. A DM is a set of attributes, $\{a_1, a_2, \ldots, a_n\}$, which is divided into three disjointed groups: Input (IN), Output (OUT), and Others (OT) attributes.

$$IN \cap OUT = \emptyset \wedge IN \cap OT = \emptyset \wedge OT \cap OUT = \emptyset \quad (1)$$

These sets of attributes are defined as:

- *IN:* specific attributes from the DS that will be linked to the input variables of the COP.
- *OUT:* attributes that describe the output data of the COP. The values of these attributes are obtained from the COP resolution.
- *OT:* specific attributes form the DS that describe additional information and that can be used to make further queries combining them with output attributes. These attributes are unrelated to the COP since they lack influence in its resolution.

The DM enables the separation of the DS and the COP model in such a way that the COP model can be applied to many data sets. However, to relate the DM and attributes of the DS and the variables of the COP, we need to introduce another concept, the DMap.

Following definitions determine how the DS attributes and the COPs inputs are mapped through the DM.

Definition 6. $\text{DMap}_{\text{DS}\rightarrow\text{DM}}$ is the relation between the attributes of DS and the IN and OT attributes of the DM, formed of a list of attributes $\{a'_i, \ldots, a'_m\} \subseteq \text{DS}$ where a subset of attributes are related to IN, $\{a'_i, \ldots, a'_m\} \mapsto \{IN\}$, and another subset of attributes are related to OT, $\{a'_h, \ldots, a'_g\} \mapsto \{OT\}$.

$$\text{DMap}_{\text{DS}\rightarrow\text{DM}} : \text{DS} \mapsto \{IN, OT\}, \quad (2)$$

$$\forall a'_i \in \text{DS} : \exists i_x \in IN | a'_i = i_x \quad \wedge \quad \nexists o_h \in OT | a'_i = o_h, \quad |IN| \geq 1 \quad (3)$$

Definition 7. $\text{DMap}_{\text{DM}\rightarrow\text{COP}}$ is the relation between the attributes of the data model DM and the variables of the COP, *V*. It is described by a list of attributes $\{i_k, \ldots, i_n, out_g, \ldots, out_l\}$ where a subset of attributes from IN are related to the input variables of the COP, $\{i_k, \ldots, i_n\} \mapsto \{V\}$, and another subset of attributes from OUT are related to the output variables of the COP, $\{out_g, \ldots, out_l\} \mapsto \{V\}$.

$$\text{DMap}_{\text{DM}\rightarrow V} : \{IN, OUT\} \mapsto V, \quad |V| \geq 2 \quad (4)$$

$$\forall i_k \in IN : \exists v_j \in V | i_k = v_j \quad \wedge \quad \nexists v_h \in V | i_k = v_h \quad (5)$$

$$\forall out_g \in OUT : \exists v_i \in V | out_g = v_i \quad \wedge \quad \nexists v_j \in V | out_g = v_j, \quad (6)$$

$$\forall (i_k, out_g) | i_k \in IN, out_g \in OUT : \nexists v_t \in V | i_k = v_t \quad \wedge \quad out_g = v_t \quad (7)$$

Following the running example, the DS in Table 1 is remarked as IN, OT, and OUT according to the DM definition and to represent the DMap. To illustrate how the Data Mapping is carried out, once the transformation has been performed. Fig. 2 represents the mapping between the DS and the COP through the DM. A first DMap determines the relation between the attributes of the DS and the DM. The second DMap relates the DM and the variables of the COPs [46]. Applied to the running example, the attributes of DS (nH, nRM, h_A, rm_A, p_A, h_B, rm_B, p_B, *Comp.*) are mapped into attributes of the DM as labelled as IN and OT in Table 2. For instance, *nH* from the DS is mapped as *nH* IN's attribute. Similarly, *nH* IN's attribute is mapped as *hoursLabour* variable of the COP.

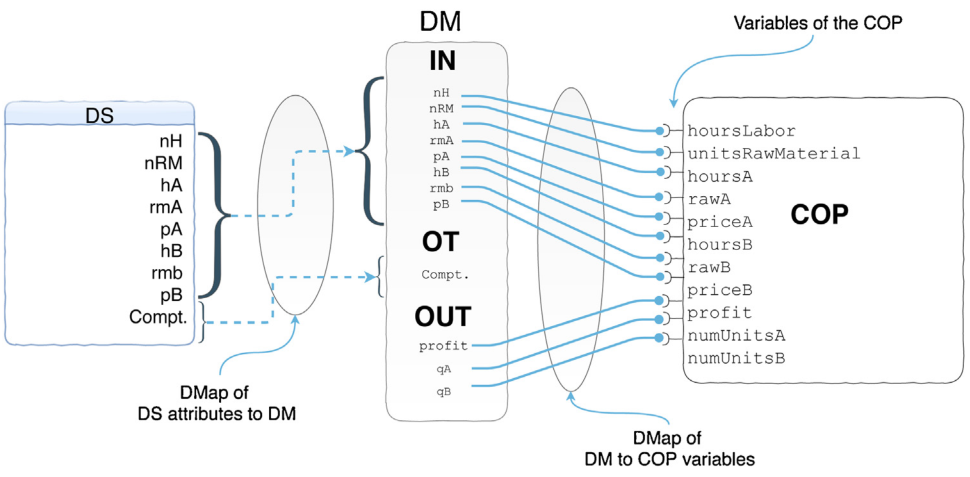


Fig. 2. Data mapping applied to the example.

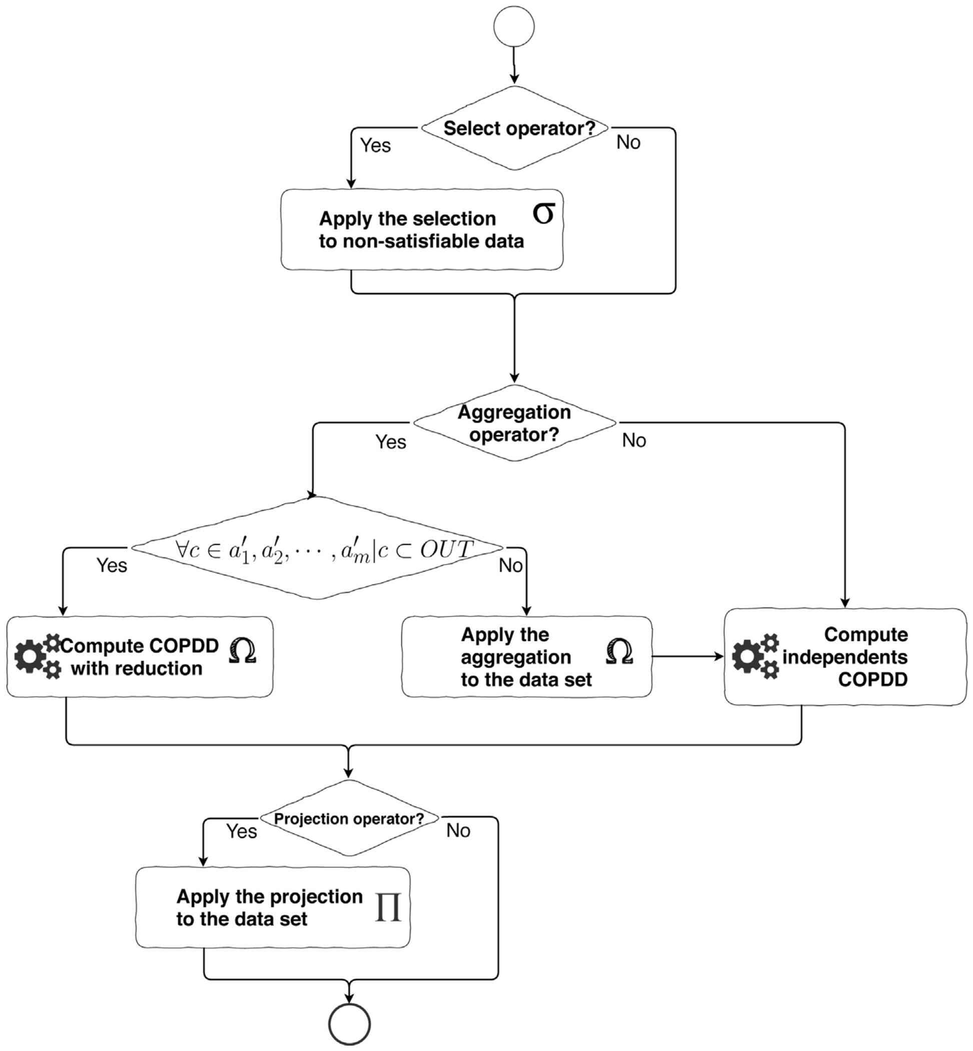

Select operator?
Yes
No
Apply the selection to non-satisfiable data
σ
Aggregation operator?
Yes
No
$\forall c \in a'_1, a'_2, \cdots, a'_m | c \subset OUT$
Yes
No
Compute COPDD with reduction
Ω
Apply the aggregation to the data set
Ω
Compute independents COPDD
Projection operator?
Yes
No
Apply the projection to the data set
∏

Fig. 3. Process for computing COPDD for distributed data.

Regarding the constraints, the COP can be modelled employing a set of fixed constraints such as seen in the example of Table 2. Therefore, each tuple represents a possible assignment of data to the variables of the COP, which can be solved (i.e., OUT) independently. However, in other types of problems, the constraints are built dynamically due to the constrained-relation existing between the values of the attributes of the DM. Thus, each tuple produces a customised COP. The solution of these customised COPs could be influenced by a criterion regarding the IN attributes, for instance, to order the COPs based on descending criteria of p_A attribute. This

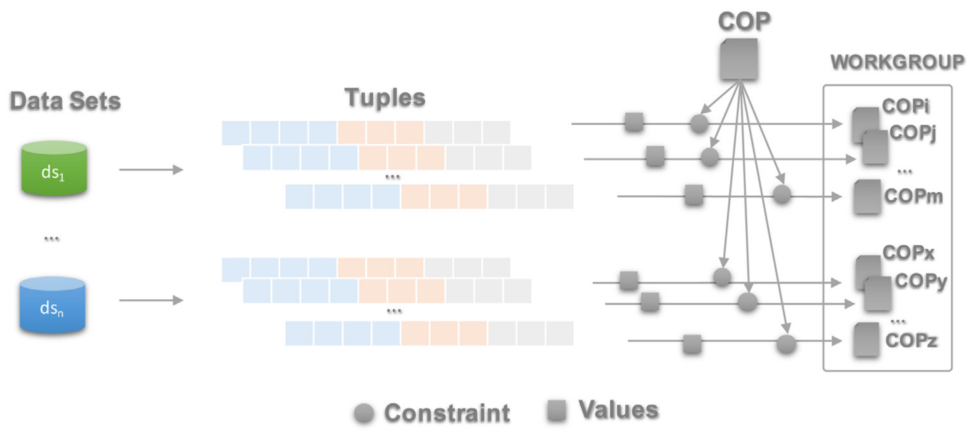


Fig. 4. Illustration for independent resolution of COPDDs.

criterion establishes the solution of *ID* tuples: # 3, # 4, # 5, # 1, and # 2. Thus, COP for the tuple # 3 will be solved in the first place, the COP for the tuple # 4 in the second place, and so on.

3.4. Data querying

The DS^{output} contains every DS tuple combined with the outputs (OUT) obtained after the distributed COPs evaluations. Later analysis of this data should be relevant. Following the running example, to know the maximum *profit*, the minimum or the average. Classical query operators can be applied to DS^{output}, but it is necessary to analyse which types of attributes (IN, OUT and OT) can be involved in each operator. For this reason, we revisit the classical relational algebra [42] employing its unary operators (i.e., selection, projection and aggregation) that can be applied to DS^{output}.

- Selection (σ). When the selection operator is applied to a DS^{output}, depending on the types of attributes involved in the φ, the selection can be applied before and/or after the resolution of the COPDD. In our proposal, only IN and OT attributes can be involved.
- Projection ($\prod$). When the projection operator is applied to a DS^{output}, the attributes involved $\{a_1, a_2, \ldots, a_n\} \subset \{\text{IN, OUT, OT}\}$. For this reason, the projection is applied after the optimisation problem resolutions, since OUT attributes can be included in the projection.
- Aggregation (Ω). There are five aggregation functions (*f*), i.e., *SUM*, *COUNT*, *AVG*, *MAX*, and *MIN*. The application of each function returns a number (Integer or Float) and can be applied in IN, OUT and OT attributes.

These three operators can be combined and nested (thanks to the closure property of the relational algebra) to query a DS^{output}. In following section, we propose to overtake part of the query evaluation to the COPs resolution to reduce the number of optimisation problems to solve, and take the advantages of the distributed and parallel evaluation of the COPs, as it is detailed in the following subsections.

4. Optimising data set query for solving COPs

The presented operators can involve every tuple, however, some data analysis done by querying could not entail every tuple, such as 'to obtain the maximum *profit* when *nH* is greater than 1, 200'. In this section, we analyse how they can be solved to reduce the computation complexity of obtaining and querying DS^{output}, proposing the systematic process shown in Fig. 3 applied before the COPDDs evaluation, since the order in which the operators are applied affects the time consuming of the distributed resolution of the optimisation problems drastically, being crucial how the tasks to solve a query are scheduled [43] in Big Data ecosystems. The application of these steps avoids the resolution of COPs that do not fit the selection operator or those that do not correspond with the attribute to aggregate.

The **first step** is to analyse whether a **selection operator** is included. It might reduce the input data involved in the COPDDs. For instance, in the running example of the previous section, let be only optimised the products whose rm_A have a value greater than 2, the tuples # 2 and # 5 might not be included in the computation of the COPDDs. If there is a selection operator, the application of the filtering must be developed, as it is carried out in classical data. In the current proposal, only selections over input and other types of attributes can be applied.

The **second step** is related to the **aggregation operator**. If an aggregation operator is included, how the information is managed depends on the types of attributes and the aggregation functions (i.e., *SUM*, *COUNT*, *AVG*, *MAX*, and *MIN*). IN and OT are managed as usual data applying the aggregation operator or relational algebra, meanwhile OUT for *MIN* or *MAX* function will be used to improve the optimisation of the COPs to reduce the search space of the variables. Therefore, the **Computation of independent COPDD** or **Computation COPDD with reduction** will be carried out depending on whether an aggregation operator over OUT attributes have been used or not. Both types of COPDDs are explained in the following subsections. The **third step** is related to the **projection operator**, where a new DS is created as a subset of attributes of the DS^{output}. This new DS is the result of the projection operator and the solution to the problem. An example is to obtain the *profit* but not being necessary to return q_A and q_B.

4.1. Computation of independent COPDD

When operators are applied to OUT attributes, every COPs can be solved individually, being or not in distributed nodes. Thereby, the DS described in the formalisation can be divided into different and distributed nodes. DS can be represented as a set of data sets, $ds_1, ds_2, \ldots, ds_n$, each of them located in an independent node. Each tuple p within a $ds_i \in \{ds_1, ds_2, \ldots, ds_n\}$ represents a problem to be optimised. Each COP depends on the values of the DM for each tuple. It is similar to use each tuple p to instantiate the COP, creating a COP$_p$ in which the values are taken from p. The group of COPs generated for each tuple of all DS conforms a *workgroup*, as we defined in Fig. 4. A particularity of this scenario is that the COPs within the *workgroup* can be computed independently without the application of any operator. Thus, the *workgroup* can be seen as a unique task

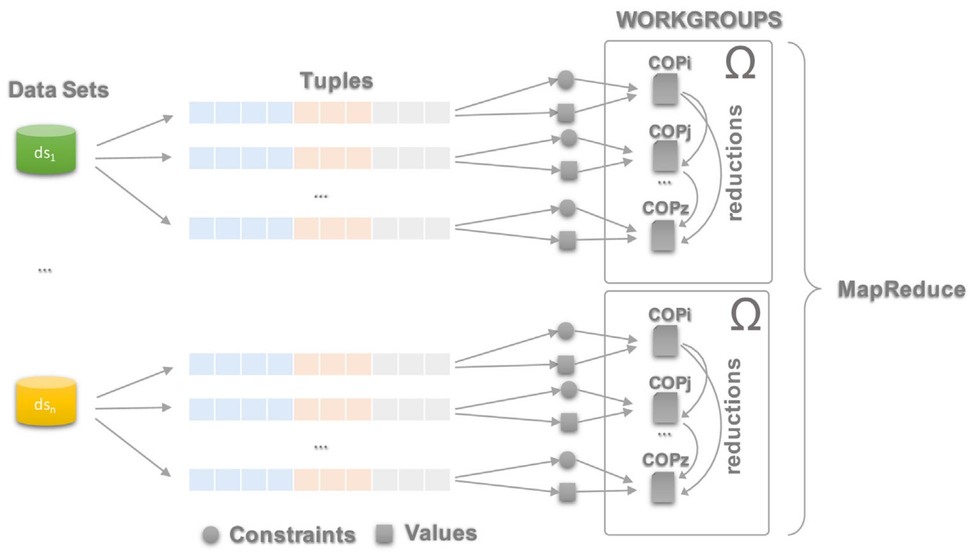


Fig. 5. Illustration for reduction based on aggregation.

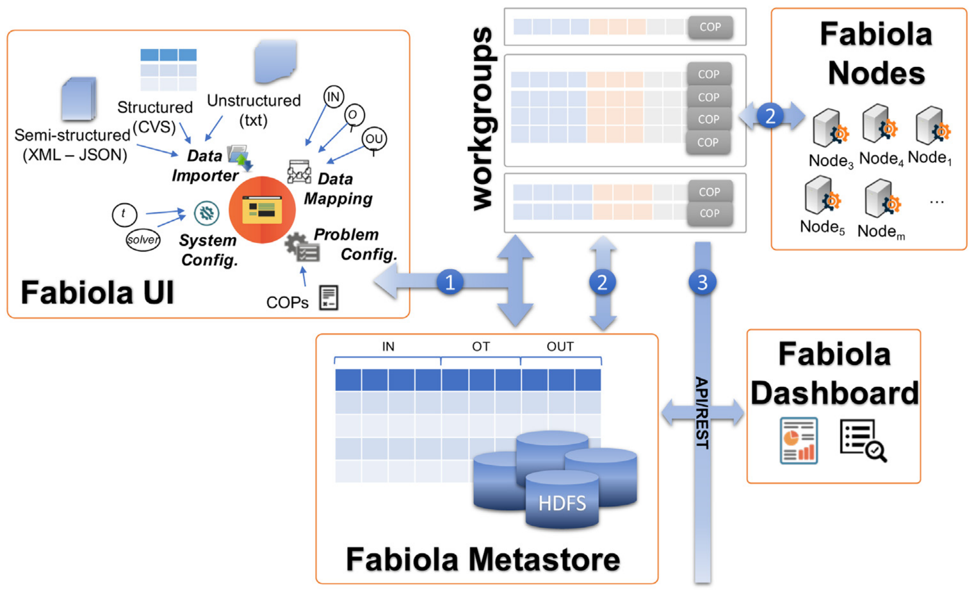


Fig. 6. Architecture and methodology of FABIOLA.

that needs to be computed in a distributed way. As commented, in the running example can be to obtain the average of the *profit* for all the products, being necessary to compute all the COPs to obtain the *profit* values before obtaining the average. The calculation of the *profit* for each COP is unrelated to any other COP. Therefore, the computation of the COPs can be performed independently at any order, because all the COPs must be solved.

4.2. Computation of COPDD with reduction

When an aggregation operator is applied to obtain the *MAX* or *MIN* value of an OUT attribute, the evaluation of a COP_p can use solutions obtained from the resolution of previous COPs of the same DS (ds_i). Thus, each tuple conforms a COP_p in which constraints and values are taken directly from *p*. There are possible values from the same DS that might be shared from one solved COP_i to another unsolved COP_j to improve the search for solutions reducing the search space. Thus, the domain of variables of the (unsolved) COP_j might be reduced to enhance the search space. Following the running example, the objective function could be to determine the maximum *profit* with the minimum value of units of A (cf., q_A). In this case, assuming the solution of the COP, for instance, for the tuple # 1 the values of *profit* and q_A (cf., Table 1) could be used to reduce the domains of the variables *profit* and $numUnits_A$ in the unsolved COPs since a greater value of $numUnits_A$ and a less value of *profit* than previously calculated are unacceptable. Despite reduction operations, the aggregation operator (i.e., Ω) is necessary since the results of all COPs must be combined to determine the maximum *profit* and the minimum q_A. As defined in Fig. 3, the *aggregation operation* has to be applied in the output variables *profit* and $numUnits_A$.

In this scenario, problems from the same DS might be related internally to each other by *reduction operations* but they are unrelated to problems from other DSs. Therefore, the COPs of each DS, ds_i, conforms independent *workgroups* that can be solved without regarding other *workgroups*. Thus, each *workgroup* can be seen as a unique task that needs to be computed in a distributed way. In Fig. 5, two *workgroups* are depicted related to the DS, ds_1 and ds_n,

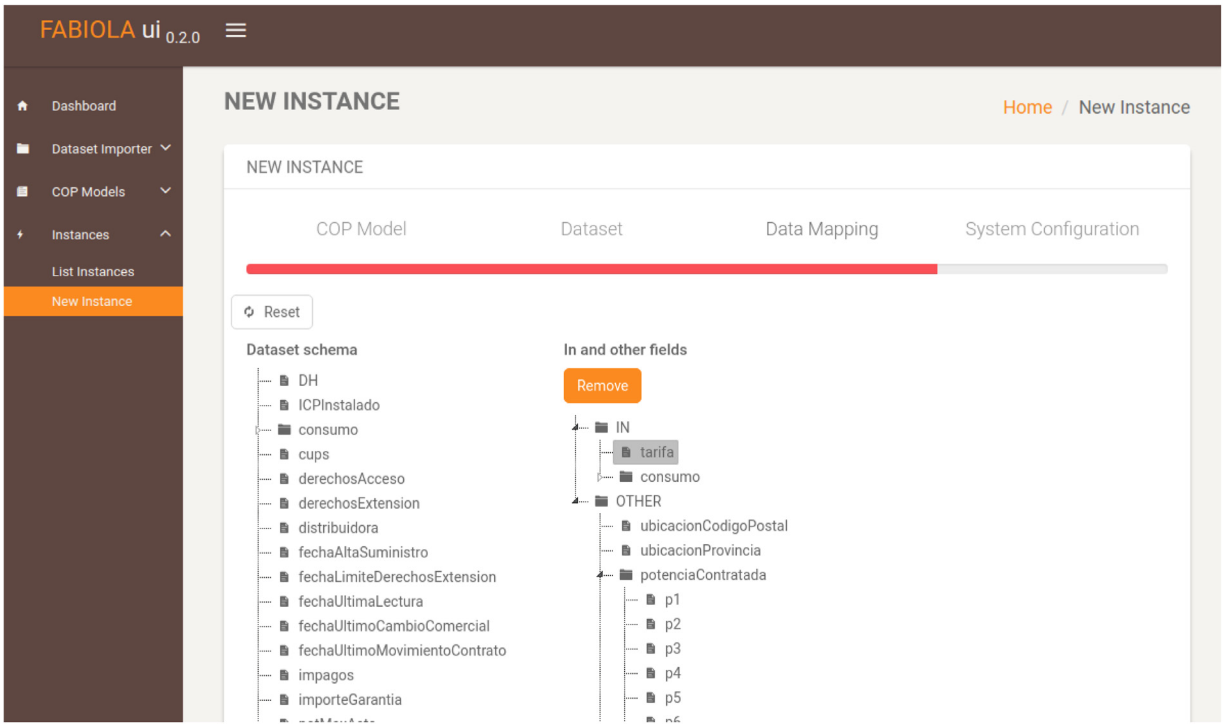


Fig. 7. Data mapping in FABIOLA-UI.

wherein each *workgroup* reduction are illustrated as arrow transitions between the COPs and Ω represents the aggregation operator which is applied in this context.

In Big Data environments, the resolution of a problem can be based on partial problem resolutions that are combined, which is known as MapReduce technique [10]. It can also be applied to the resolution of optimisation problems. For instance, the maximum value of an attribute, where the value of this is distributed in different *workgroups*, is the maximum value of the all maximum obtained for each *workgroup*. This implies to optimise a problem, with the optimisation of each *workgroup*, that has been solved in proposals such as Hive [45]. The question is, how this optimisation process can be affected when the partial resolution implies also the COP resolutions. The solution of all *workgroups* need to be combined (cf. Fig. 5) to generate a global solution as a new OUT. A way of combining needs to be defined over the OUT attributes of COPs. In order to carry out this, MapReduce for optimisation is defined as follows.

Definition 8. A MapReduce for Optimisation Problems, *MR*, is defined as a combination function which establishes the maximum or minimum value of an OUT_i variable, being $OUT_i \in \{out_1, out_2, \ldots, out_n\}$ and establishes a way of the combination by means of the operator, OP, as minimise (*MIN*) or maximise (*MAX*).

Each variable $v \in \{out_1, out_2, \ldots, out_n\}$ involved in the aggregation operator for a COP_i is bounded with a specific value obtained in previous solved COPs of the same DS, (ds_j). This specific value is a maximum whether the *MIN* value wants to be obtained, or the minimum whether the *MAX* function is used in the aggregation.

5. FABIOLA: the technological approach

FABIOLA is presented as a technological solution for the automatic creation of COPs with distributed Data. In this section, the needed infrastructure, including the architecture, and the implemented tool are detailed.

FABIOLA as a technological solution is composed of two differentiated parts: (1) the infrastructure (i.e., back-end) for the computation, and; (2) a web interface (i.e., front-end) to enable the access to the components described in previous sections. The FABIOLA back-end and front-end components are depicted in Fig. 6.

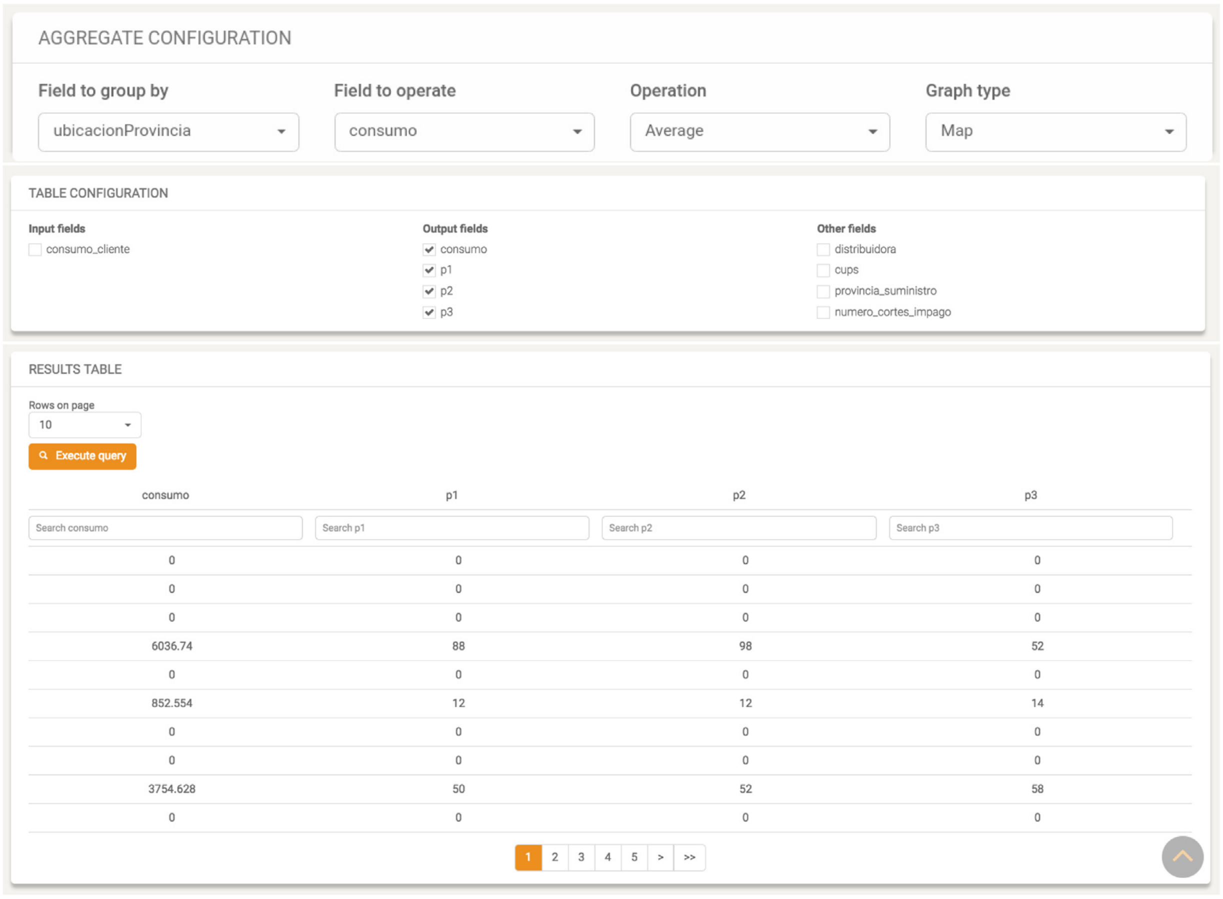
The FABIOLA front-end is divided into two components:

- **FABIOLA-UI** component is a web client-side application which enables to setup the environment to compute COPDDs. It provides sections for the *Data importer* to load external data; the COP description; Data Transformation; the *Problem configuration* (i.e., *Data Mapping*), and the *System Configuration* in terms of solver, memory, timeout of execution, etc.
- **FABIOLA Dashboard** is a control panel from where the user can use various reporting and querying components. The dashboard enables users an easy-querying and visualisation of the data and results in FABIOLA.

FABIOLA-UI, on the left, provides a fixed menu which gives direct access to different parts of FABIOLA components such as the dashboard, DS importers, DS creation, COP models, and instances as shown in Fig. 7. The same figure presents the main view once entering FABIOLA-UI in the section of creating a new instance. The creation of a new instance in FABIOLA means the creation step-by-step of a new COPDD. The figure shows an example of how to carry out the data mapping between the DS (cf., DS schema) and the attributes of the DM. The user can choose by drag-and-drop which attributes from imported DS are IN and OT.

FABIOLA-UI provides an editor for the description of the COP model in Constraint Optimisation Problem. Currently, FABIOLA-UI supports any ChocoSolver-based optimisation syntax. Although the description of the COP can be a difficult task, it is done once and can be applied to every DS. In a similar way than in data mapping between DS and DM, the DMap between DM and the variables of the COP is described.

FABIOLA-UI provides customised interfaces where operators can be applied over the data and COP results. For instance, *projection operations* can be applied for the results such as the forms provided. In this case, it is described the attributes to projection, selection and



AGGREGATE CONFIGURATION

Field to group by	Field to operate	Operation	Graph type
ubicacionProvincia	consumo	Average	Map

TABLE CONFIGURATION

Input fields	Output fields	Other fields
☐ consumo_cliente	✓ consumo	☐ distribuidora
	✓ p1	☐ cups
	✓ p2	☐ provincia_suministro
	✓ p3	☐ numero_cortes_impago

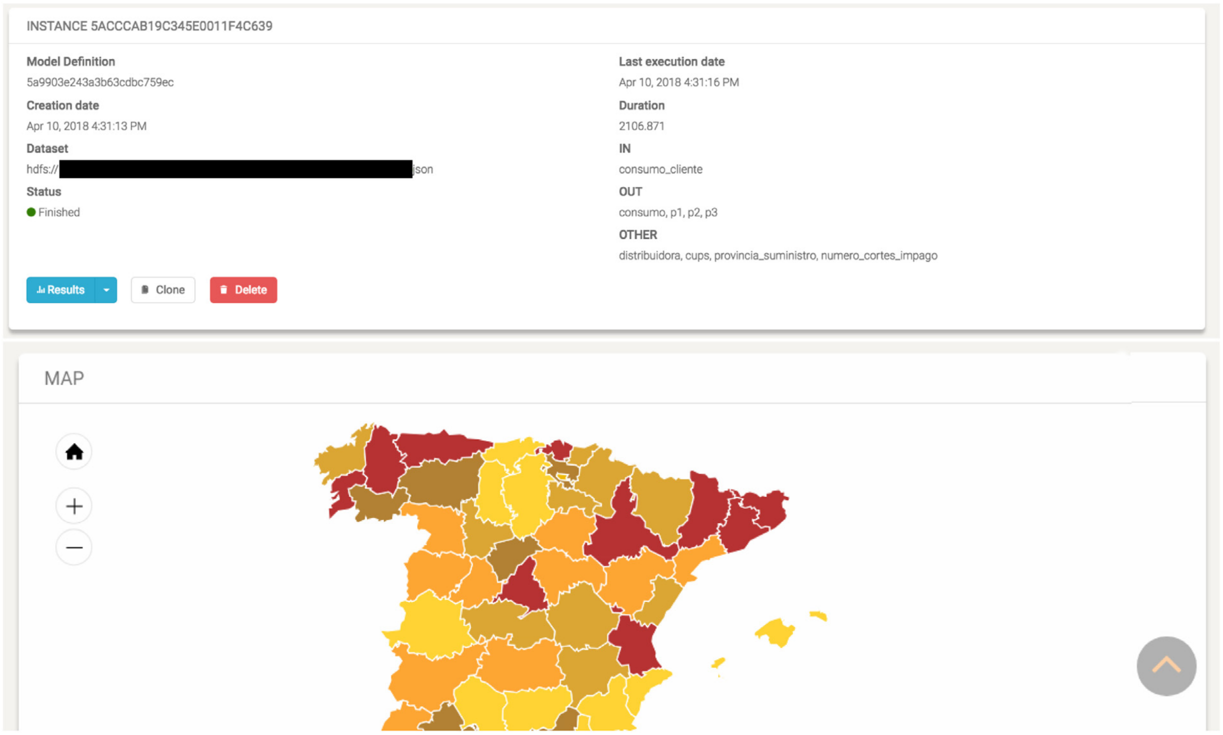
RESULTS TABLE

Rows on page: 10

Execute query

consumo	p1	p2	p3
0	0	0	0
0	0	0	0
0	0	0	0
6036.74	88	98	52
0	0	0	0
852.554	12	12	14
0	0	0	0
0	0	0	0
3754.628	50	52	58
0	0	0	0

Fig. 8. Data visualisation of results.



INSTANCE 5ACCCAB19C345E0011F4C639

Model Definition
5a9903e243a3b63cdbc759ec

Creation date
Apr 10, 2018 4:31:13 PM

Dataset
hdfs://[redacted]json

Status
● Finished

Last execution date
Apr 10, 2018 4:31:16 PM

Duration
2106.871

IN
consumo_cliente

OUT
consumo, p1, p2, p3

OTHER
distribuidora, cups, provincia_suministro, numero_cortes_impago

Fig. 9. Data visualisation aggregated by regions.

aggregation, as shown in Fig. 8 respectively. This use of operators will permit the view of the data, enabling an easy data visualisation and analytics where grouping and aggregated operators can be applied for filtering the results.

Other ways of presenting, grouping and aggregating of results can be defined. Fig. 9 presents an example of a dashboard for showing the results and the information about the computation of the COPDD. In this case, the results have been ranged in regions and located in a geographical map since FABIOLA Dashboard lets the extension by using modules to present data results.

The FABIOLA back-end is composed of the following components:

- **FABIOLA Metastore** represents the internal data storage mechanism of FABIOLA. It presents the problems in a structured way, where data is organised in the form of tables and schemes, such as in Hive [45]. These tables and schemes are only a virtual representation (i.e., view) of the data since it is internally organised in the original format of a distributed file system, for instance, HDFS or NFS.
- **FABIOLA Nodes** are responsible to compute the COPs. Thus, each row (tuple) instantiates a COP, and the generated COPs are grouped into *workgroups*. These *workgroups* are uniformly allotted to be solved among the available nodes and according to the workload of the nodes.

The FABIOLA back-end has been implemented by an Apache Spark [49] cluster managed by DC/OS.[2] The cluster consists of (cf., Fig. 10) a master node, responsible for planning the execution and managing the cluster resources, and N agent nodes, which are responsible for managing the applications deployed on the cluster. In our case, the cluster is composed of five agent nodes for running Apache Spark. One of these nodes is the driver, responsible for creating and distributing the tasks. These are executed by the four Spark workers. Additionally, the architecture includes a server with HDFS (cf., HDFS [3]) to store the DSs, and a database (cf., MongoDB [30]) for storing execution and partial results. Regarding computational characteristics, each node from the cluster reaches four cores and 16 gigabytes of main memory.

6. Computational experiments in a real scenario

In this section, the application of FABIOLA on a real case with and without the data set query optimisation is applied (cf., Section 4) is illustrated in depth.

6.1. Case of study: electric energy consumption scenario

In Spain, seven wholesale electricity companies are responsible for supplying power to all users. However, more than 250 distributors are responsible for selling power directly and invoice for it. Customer acquisition is, undoubtedly, the most important target that the distributors have. This acquisition determines the success or failure of the company. If an electricity retailer optimises its customers' invoice, its client portfolio will increase. The optimisation of the invoice includes the customisation and improvement according to the specific client consumption. In this case, this scenario is focused on the optimisation and customisation of the consumption of each point of supply but using the rates and prices established by the Ministry of Industry. In this scenario, it is necessary to build a COP, which is going to be applied to each of the specific values of consumption of each of the existing supply points (more than 20,000,000 points in Spain). The computer processing is even more difficult since the information provided by the seven wholesalers varies in format and is distributed in various servers.

Although the format of the data sets provided is heterogeneous, the DSs are composed of a set of power invoices belonged to several users. Each power invoice is at the same time consisting of a set of month invoices belonged to a specific user. Each month invoice has information about: customer's contact details, invoice issue date, customer ID (identification number that identifies a user through a supply point at a specific address), invoice number, services hired, and consumption details. There is three types of power: active, reactive, and apparent power. The users contract a specific amount of each kind of power, and then, they consume another amount (more power or less power). Therefore, the consumption details are determined by the hired and the consumed power.

The objective is to determine the minimum cost with regards to the consumption of each customer being necessary to create thousands of optimisation problems, one of each customer. For example, which is the minimum cost for a customer that consumed a specific amount of power during a year.

6.2. Specification of DM, DMap, COP, operators, and system configuration

The DS is formed of IN and OT attributes, according to the mapping with the DM. More attributes can also participate in the DS, but in the used example, all attributes are IN or OT. Following is included the description of each DS attribute:

- **Specification of IN and OT of the DM and the** $\mathrm{DMap}_{\mathrm{DS}\rightarrow\mathrm{DM}}$**:**
  –**IN attributes of the DS:**
  * *CustomerID*: customer identification that uniquely defines a customer. It is the ID of the person.
  * *PowerConsumption*: power consumption over a period such as twelve months or more. Each period is defined by a triple such as $\langle p1, p2, p3\rangle$, for instance $\{8.0, 8.0, 10.0\}$ that represents the power consumption in *kwh*.
  * *Period*: the period in which each record of the DS was captured, such 365 as the number of days of information stored.
  * *TariffHired*: rates hired by the customer.

  –**OT attributes of the DS:**
  * *PostalCode*: location for the customer service, for instance, zip code.
  * *CustomerContactDetails*: extra contact information about the customer.
- **COP**: the constraints of the problem are imposed by the local regulation [20] depending on the rate and power consumption hired by a customer. To illustrate the case, a piece of the code of constraints is included in Fig. 11.
- **Specification of the IN and OUT of the DM and the** $\mathrm{DMap}_{DM\rightarrow\mathrm{COP}}$:
  –**IN attributes of the Data Model related to the COP:**
  * *CustomerID* of the DM is related to the variable *customer* of the COP.
  * *PowerConsumption* of the DM is related to the COP variable *currentConsumption*.
  * *Period* of the DM is related to the variable *comsumption. days* of the COP.
  * *TariffHired* of the DM is related to the variable *Price*.
- **Specification of the OUT attributes of the DM related to the COP:**
  –*EstimatedCost*: optimised cost, this data should improve the real price. This attribute is mapping to the variable *TotalPrice* of the COP.
  –*EstimatedPower*: the optimised power which improves the real power consumption. This is given by triple $\langle p1, p2, p3\rangle$ similar

[2] DC/OS by D2iQ: `https://dcos.io/`.

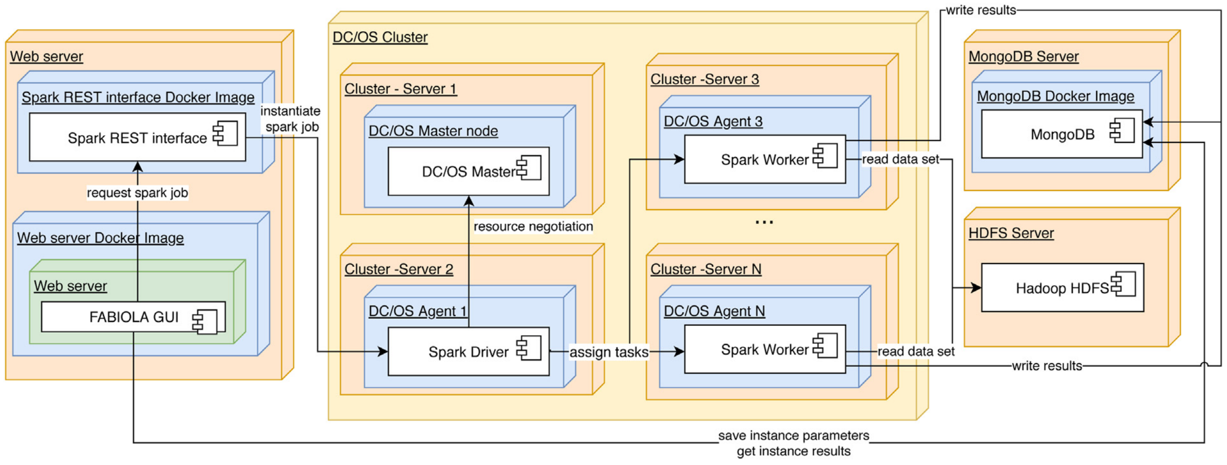


Fig. 10. FABIOLA cluster architecture.

```
Variables and Domains: {
   Var custormer = String;
   Var CurrentConsumption[12][3] = IN.powerconsumption;
   Var Days[12][3] = IN.consumptions.days
   Var Price[3] = IN.TariffHired
   Var TotalCost = intVar(0,MAX_INT_BOUND);
   Var HiredPower[3] = intVar(0,MAX_INT_BOUND);
   Var PowerInvoice[12][3] = intVar(0,MAX_INT_BOUND);
   Var PowerTerm[12][3] = intVar(0,MAX_INT_BOUND);
   Var TotalPrice = intVar(0,MAX_INT_BOUND)}
Constraints: {
   forEach(i <- 0 until 12, j <- 0 until 3) {
     if(HiredPower[j] * 85 > CurrentConsumption[i][j] * 100)
     then(100 * PowerInvoice[i][j] - 85 * HiredPower[j] = 0)

     if(HiredPower[j] * 105 < CurrentConsumption[i][j] * 100)
     then(100 * PowerInvoice[i][j] 210 * HiredPower[j] = 300 * CurrentConsumption[i][j])

     if(HiredPower[j]  *  85  <  CurrentConsumption[i][j]  *  100  AND  HiredPower[j]  *  105  >=
        CurrentConsumption[i][j]* 100)
     then(100 * PowerInvoice[i][j] = 100 * CurrentConsumption[i][j])

     PowerTerm[i][j] = PowerInvoice[i][j] * Price[j] * Days[i][j]
   }
   forEach(i <- 0 until 12) {
     TP[i] = Sum(PowerTerm[i])
   }
}

Objective Function: {
   TotalPrice = Sum(TP)
   MINIMIZE(TotalPrice)
```

Fig. 11. COP for the optimisation of customer cost.

to the *PowerConsumption*. These attributes are mapping to the variable *HiredPower* of the COP.

- **System Configuration:** the configuration of the system will depend on the characteristics of the problem, but some configurations must the described mandatory in every problem:
  –**Node Master cores**: one driver is needed at least.
  –**Node Master memory**: at least one gigabyte for the master node is needed.
  –**Node Executor core**: the number of executors is determined automatically regarding the size of the job, the available resources, and the load of the cluster. However, one executor at least will be used.
  –**Nodes Executor memory**: the memory should be tuned in function of the load of the problem, in this particular case with four gigabytes at least per executor is enough.
- **Application of Operators:** the use of operators can help to extract the needed information, for instance, some queries for the example are:
  –*Which is the region where the power consumption is minimum or less than 400 kWh per month?* This question implies at an *aggregation* of solutions per regions and the application of a MapReduce of solutions to determine the minimum. Moreover, the solutions obtained probably will require some *projections* to represent the results as customer required.
  –*Which is the average of power consumption of customers with an A3.0 hired tariff per regions?* This question implies a *selection* to manage only the tariff A3.0, a *projection* of per hired tariff afterwards an *aggregation* which enables the average consumption per customer.
  –*Which is the average of optimisation in a specific region?* First, the optimisation is determined per customer without an oper-

Table 3
Data set information.

Data set	Size (MB)	Number of users
DS1	2058.98	530,504
DS2	441.2	37,279
DS3	2902.29	373,701

ator, afterwards an *aggregation* of those results per region is needed to determine the average of optimisation per region.

6.3. *Empirical evaluation and results*

In order to evaluate our proposal, several benchmarks have been carried out. The benchmarks have been run over three real DSs (DS1, DS2, DS3) of power providers of Spain. Each DS consists of information respect to the power consumption of customer for several years with the data presented in the previous subsection. The information about the size and the number of customers that includes each DS are given in Table 3.

We have developed two benchmarks whose objectives are:

- *Benchmark I*: *Which is the optimised power consumption to improve (i.e., reduce) the cost of all customers?*
- *Benchmark II*: *Which is the best profit regarding the optimised cost?*

Thus, *Benchmark I* pursues the optimisation of the power consumption of all customers to reduce their costs, and *Benchmark II* pursues the determination of the best (i.e., minimum cost) profit in terms of improvement (i.e., maximum) of the estimated cost. The *Benchmark I* is based on the idea of Computation of independent COPDD (cf., Section 4.1) since the COPs are independent and any operators are required to combine solutions, while *Benchmark II* is based on the Computation COPDD with domain reduction (cf., Section 4.2) since operators of *selection, aggregation and projection* could be required.

6.3.1. *Benchmark I: computation of independent COPDDs*

Regarding the computational impact, this benchmark aims to demonstrate the impact of the execution of COPs in stand-alone (i.e., sequentially in just one node) and using FABIOLA (i.e., parallelised). The COPs to be enacted in both cases are the same, therefore, they are comparable and compatible. This comparison is performed by evaluating the scalability as the number of COPs increases by means of asymptotic analysis.

For this study, DS1, DS2 and DS3 are employed. For each one, an analysis of the scalability of our proposal is performed. This analysis is carried out by progressively increasing the size of each DS. For instance, DS1 is scaled five times. Then, five executions are performed for each one, and the average Elapsed Real Time (ERT)[3] is calculated. This will enable us to analyse the temporal complexity from the user's point of view as the number of problems increases. This test is performed in sequential as well as in parallel.

Regarding the configuration of the experiment, all the resources of the cluster previously described are employed. It means that there are four Spark workers for solving the COPs in parallel, each one provisioned with four cores and 16 gigabytes of main memory. On the other hand, the *timeout* is fixed to 5 s.

Although several metrics have been captured in the test, we have them grouped in three types of metrics: (i) problem modelling; (ii) execution performance (in time and memory), and; (iii) impact of *timeout*.

[3] The ERT is the time from the start of a program to its end.

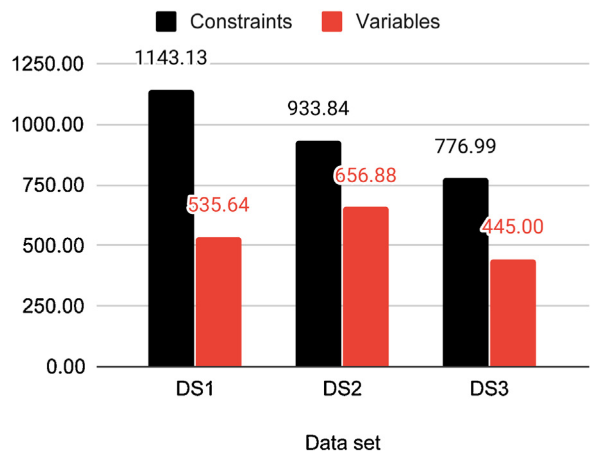


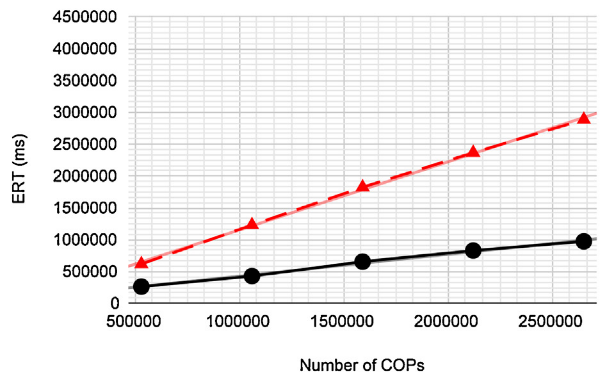
Fig. 12. Number of constraints and variables.

Regarding **problem modelling** metrics, for each DS, the number of constraints and variables resulting from the COP model is quantified. The number of constraints and variables depends on the attributes of the DS and the solver. The solver will require instantiating internal constraints and variables. Note that the average number of constraints and variables is not affected when proportionally scaling the DS, since the proportion will always be the same. Moreover, it is not affected by whether the execution is sequential or parallel. It only depends on the DS, and hence, these results have been depicted by DS. These are depicted in Fig. 12.

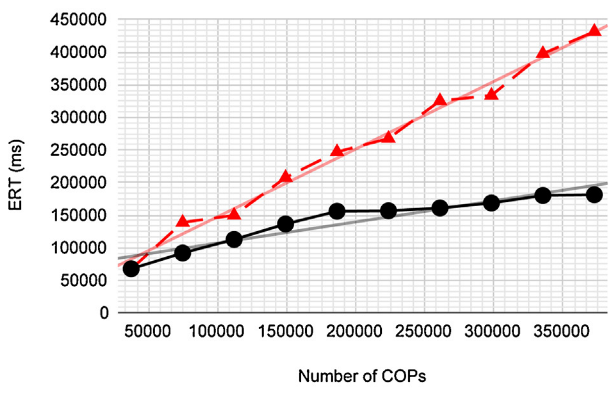
From the results in the figure, it is possible to conclude that the complexity of DS3 might be minor than the other two DSs, since it has the smallest number of variables. On the other hand, it is impossible to ensure that the complexity of this DS is less since there are other factors that can condition the complexity, for instance, constraints can affect the complexity of the COPs. The more constraint, the more the domain is limited, hence, a greater number of constraints would imply (in theory) less complexity. Albeit to ensure that, every single constraint would have to be analysed in order to see how it affects the domain of the variables and it is not a trivial task.

Regarding the **execution performance** metrics, asymptotic analysis is carried out. Fig. 13 depicts the results of these tests for the three DSs, showing how the ERT varies as the number of COPs increases. While DS1 and DS3 have been scaled five times, DS2 has been scaled 10 times. It is due to the fact that DS2 is smaller than DS1 and DS3, and scaling it just five times is not enough to compare it to the other two DSs. In addition, the behaviour in the first five tests in parallel differs from the behaviour in the last five. Note that the ERT tends to converge after the fifth point (i.e., after scaling the DS five times). On the other hand, the execution of DS2 without scaling it (cf., first black point in the chart (b)) yields better results sequentially than in parallel. From that point, the execution in parallel improves as the size of the DS increases. It is due to the fact that the size of the DS is very small, thereby it is not worth executing it in parallel since to distribute the COPs among the cluster nodes involves a high cost compared to the resolution time.

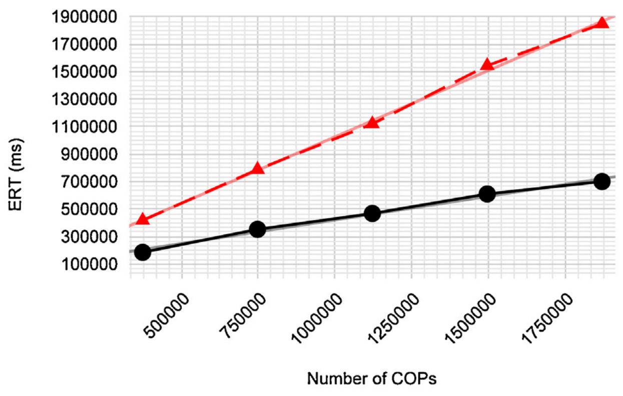
As can be appreciated, the sequential ERT is worse than the parallel. In addition, the executions in parallel tend to behave better as the size of the problem (i.e., the number of COPs) increases. In order to analyse the limiting behaviour of our proposal (i.e., an asymptotic analysis), the trend-lines corresponding to each set of executions has been calculated in Table 4. The equations of the trend-lines are shown. In these equations, y stands for the dependent variable (i.e., the ERT), while x stands for the independent variable (i.e., the number of COPs to solve).



(a) Results of ERT for DS1.



(b) Results of ERT for DS2.



(c) Results of ERT for DS3.

Fig. 13. Each point and triangle are the ERT average for five executions. Red and black line represent the sequential an parallel execution respectively.

Table 4
Trend-line equations for the execution performance of sequential and parallel solutions per DS.

Data set	Sequential	Parallel
DS1	$y = 1.07x + 84245$	$y = 0.34x + 84569$
DS2	$y = 1.04x + 43887$	$y = 0.32x + 74748$
DS3	$y = 0.97x + 59582$	$y = 0.34x + 77845$

The equation slopes are similar for all the DSs. It means that the differences between the complexities of the three DSs do not highly impact the performance. As can be observed, the slope is approximately 1 for the sequential executions. It implies a linear complexity (i.e., $\mathcal{O}(n)$). Regarding the parallel execution, the slope is approximately 3 times less than the linear slope. Hence, the complexity is sub-linear (i.e., $\mathcal{O}(\frac{n}{3})$). From these results, it is possible to conclude that the execution of COPs in parallel improves the sequential execution. Remark that, as observed with DS2, the parallel execution is not always worth. There is an intersection point from which the parallel execution improves the sequential. For DS1, this point is reached at 444; for DS2, the intersection is reached at 43, 102, and; for DS3, it is reached at 29, 314. For a number of COPs less that those values, the parallel distribution is not worth.

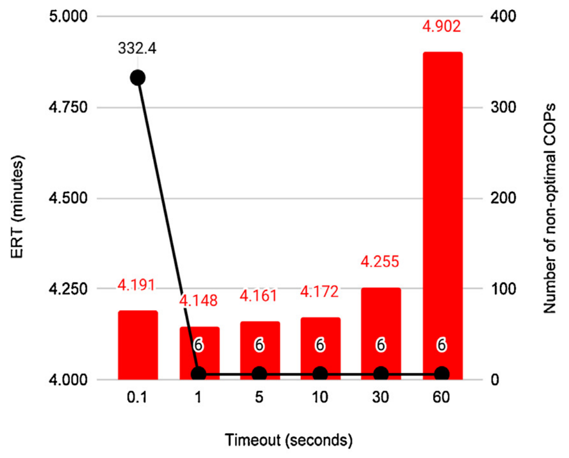
Regarding the **impact of the timeout**, six different values for the *timeout* have been studied on the DSs: 0.1, 1, 5, 10, 30, and 60 s. Thereby, for each *timeout*, five executions are performed, and the average ERT and number of non-optimal COPs are calculated. While the ERT is expected to increase as the *timeout* increases, the number of non-optimal COPs is expected to decrease.

Fig. 14 shows the results for this study. Regarding the ERT, it increases when as the *timeout* increases except for small *timeout* values. It is due to the fact that small *timeout* values do not affect the execution, especially when the number of non-optimal COPs is small compared to the size of the dataset. On the other hand, larger *timeout* values do cause an impact on the ERT. In this case, the few COPs that have not been optimally solved (approx. 6 for DS1 and DS3) are creating bottlenecks.

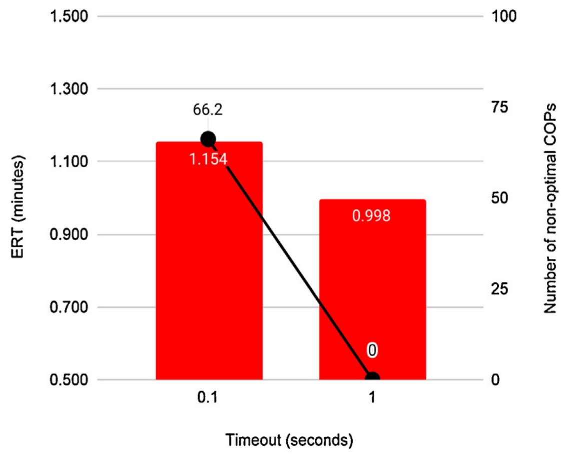
Remark that for DS2, all COPs were optimally solved with a *timeout* of 1 second. For this reason, other timeout values have not been checked since the behaviour would be similar. Contrary to the expected, the ERT when the *timeout* is 1 second. The previous reasoning applies here. These values of timeout are so small that COPs will not behave as bottlenecks.

These results support the use of five seconds as *timeout* for the tests that have been carried out in this section since it offers a good balance between the ERT and the number of COPs that are optimally solved.

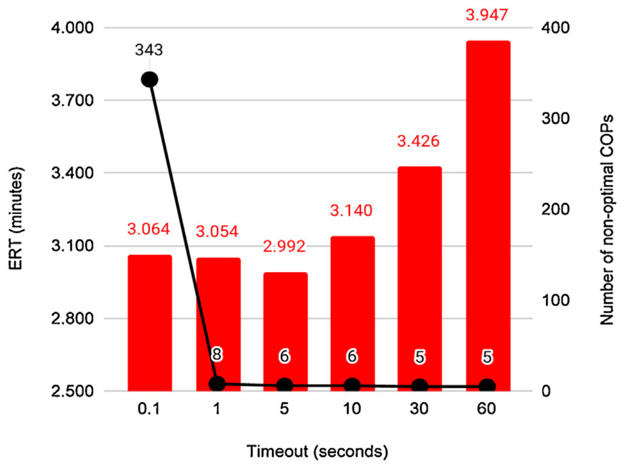
Regarding the **memory performance**, it depends on how the program is planned and the distribution of the workload among the nodes. In this case, the program is composed of two phases. While the first one consists of reading the data set from the data sources, the second phase is based on building and solving the COPs. Then, the aggregated amount of memory approximate linearly to the size of the data set independently whether or not the execution is sequential or parallel as shown in Fig. 15. This phenomenon is due to the initial read and distribution, the data to be processed is not distributed among the nodes of the cluster. Remark that all of them present an upper linear slope, the differences among them are due to the differences in the average size per COP in each DS. For instance, the average COP size is 1.2×10^{-5} KB for DS2, 7.8×10^{-6} KB for DS3, and 3.9×10^{-6} KB for DS1.



(a) Timeout tests results for DS1.



(b) Timeout tests results for DS2.



(c) Timeout tests results for DS3.

Fig. 14. Behaviour of the ERT and number of non-optimal COPs as the timeout increases. Red columns represent the average ERT of five executions, and black lines represent the average number of non-optimal COPs of five executions.

6.3.2. Benchmark II: computation COPDDs with domain reductions

All the metrics in the previous section can help to understand the problem in the form of the COPs and the impact of parallelisation in the resolution of COPs. In the Benchmark II, we want to demonstrate other aspects related to the application of operators and their impact in the time of solution.

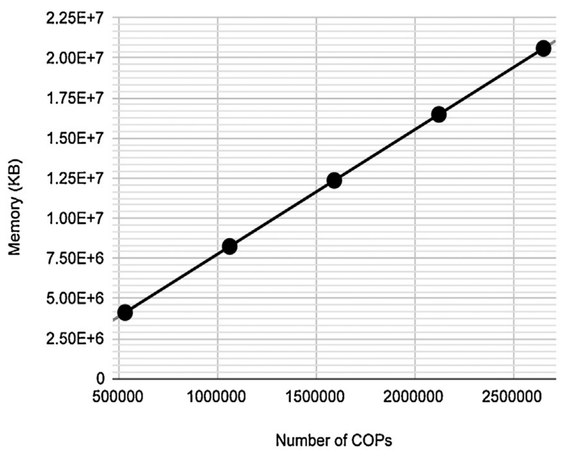
This benchmark pursues the determination of the best profit (i.e., minimum cost) in terms of improvement (cf., maximum) of the estimated cost. First, an *aggregation operator* is applied to range the customer from low to the high hired tariff. Afterwards, the customers are split into *workgroups*. The *reduction operation* is applied since the minimum cost is pursued the previously estimated cost is used to feed the unsolved COPs into the same *workgroup*. When all the customers have to be processed the minimum of each *workgroup* have to be combined (cf. MapReduce) in order to determine the global minimum.

In terms of the COP resolution, the objective is the reduction of the domain of the variables, by introducing an upper bound to the objective variable.

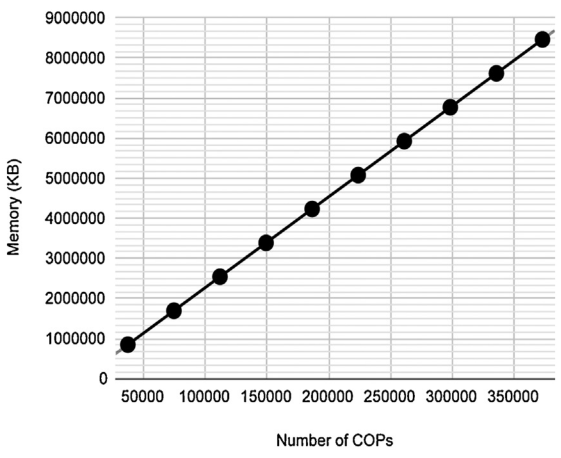
This experiment is similar to the asymptotic analysis performed in Benchmark I, but only parallel executions have been carried out. Fig. 16 depicts the results of this study, and Table 5 shows the trend-line equations for each DS. As can be seen, the aggregation operator not only outperforms the execution without such operator, but worsens the scalability (cf., note that the slopes are 0.54, 0.57 and 0.76 for DS1, DS2 and DS3, respectively). Nonetheless, by comparing the COP resolution stages of the execution, it can be checked that the reduction of the domain improves both the run-time and the scalability, except for the two first executions of DS2. It might be due to the fact that those points represent small data sets and the improvement is not noticeable by limiting the domain when running in parallel. Regarding the scalability, for DS3 the slope is slightly higher when applying reductions, which might

Table 5
Trend-line equations for the execution performance per DS.

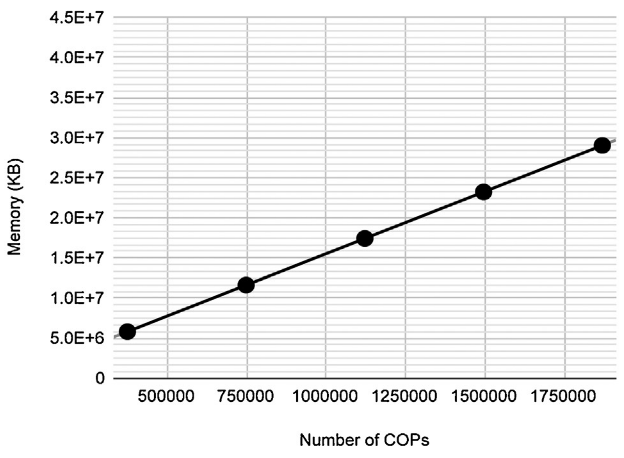
Data set	Aggregation and COP resolution	COP resolution after reduction	COP resolution without reduction
DS1	$y = 0.54x - 23300$	$y = 0.28x + 141$	$y = 0.31x + 64862$
DS2	$y = 0.57 + 101583$	$y = 0.14x + 56593$	$y = 0.24x + 56840$
DS3	$y = 0.76x + 9763$	$y = 0.27x + 750$	$y = 0.26x + 65171$



(a) Memory consumption for DS1.
($y = 7.77x - 272$)



(b) Memory consumption for DS2.
($y = 22.7x + 606$)



(c) Memory consumption for DS3.
($y = 15.5x - 91.8$).

Fig. 15. Memory consumption as the number of COPs scale progressively. Each point represents the average of the total of five executions (Kbytes).

be due to the fact that the domain reduction for this data set is not very significant.

To sum up, the application of aggregation operators does not improve the execution performance in terms of time, but it might improve the COP resolution stage of the execution depending on the nature of the data. However, this improvement might not be significant since the aggregation stage takes from the 40% to the 50% of the execution time, and also scales worse as the number of COPs increases.

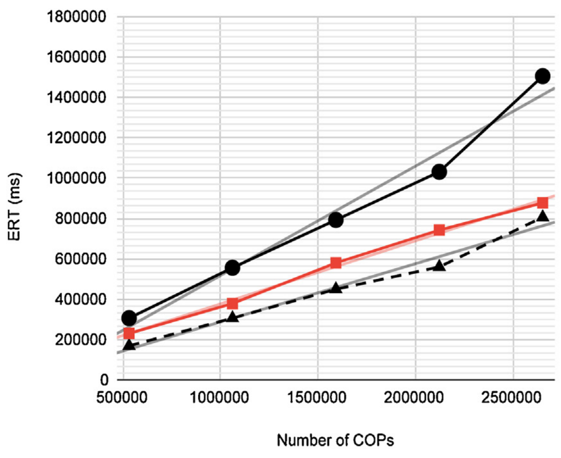
Respect to the **percentage of optimised** COPs, when applying this aggregation operator there is a high amount of COPs which are not optimally solved. Note that reducing the COP domains might lead to unfeasible problems. Hence, the proportion of solved COPs respect to the all possible COPs are 12.11% for DS1; 12.38% for DS2, and 24.9% for DS3.

Regarding the **memory performance**, in this case the program is composed of three stages. The read phase from the data sources and the COP resolution phase are the similar as Benchmark I, but between those stages another phase is included which consists of aggregating the data. This operation implies an extra memory consumption, since the records from the data set must be redistributed among the nodes of the cluster. Fig. 17 shows these results. Since applying aggregation operators require more memory consumption, these outperform the previous results. It can be checked both in the values of memory consumed and in the slope of the trend-lines, which are higher than in the previous case, implying a worsening in the scalability.

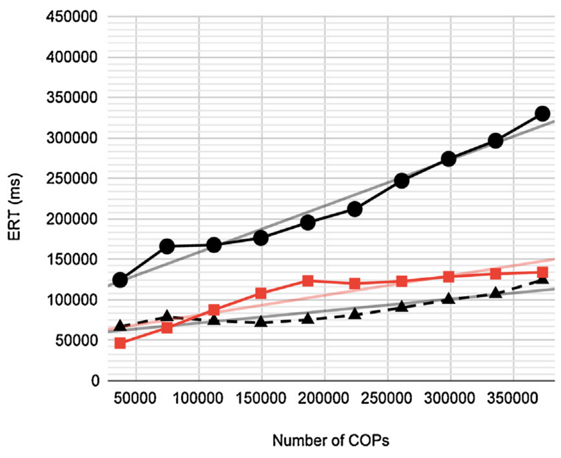
7. Related work

Big Data faces up new challenges [25,31] in how to carry out optimisation problems with heterogeneous, incomplete, and uncertain data, besides, to immediate responses for some types of questions.

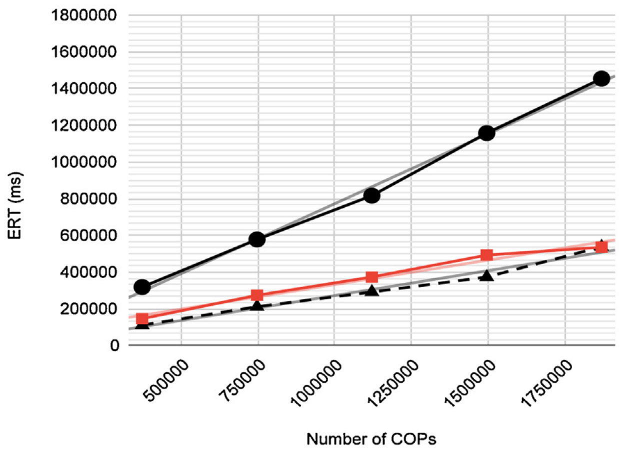
The Apache Hadoop project [3] actively supports multiple projects to extend Hadoop's capabilities and make it easier to use. There are several excellent projects to helping in the creation of development tools as well as for managing Hadoop data flow and processing. Many commercial third-party solutions build on their developed technologies within the Apache Hadoop ecosystem. Spark [49], Pig [32], and Hive [45] are three of the best-known Apache Hadoop projects. All of them are used to create applications to process Hadoop data. While there are a lot of articles and discussions about which is the best one, in practice, many organisations



(a) Results of ERT for DS1.



(b) Results of ERT for DS2.



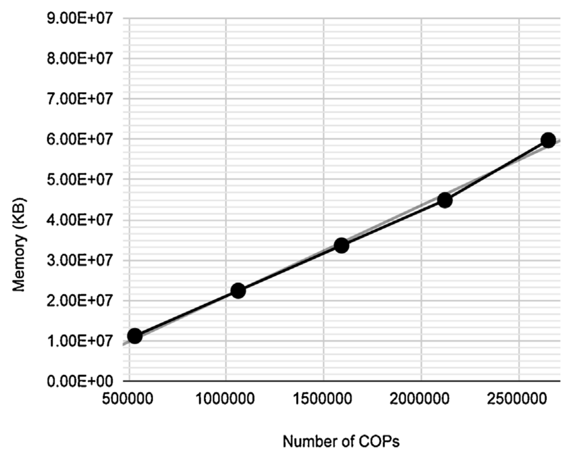
(c) Results of ERT for DS3.

Fig. 16. Each point, triangle and square is the Elapsed Real Time average for five executions. Black continuous line represents the overall Elapsed Real Time of the execution including aggregations and COP resolutions; black pointed line represents the Elapsed Real Time of the COP resolutions with domain reduction, and red line represents the Elapsed Real Time of the COP resolution without domain reduction.

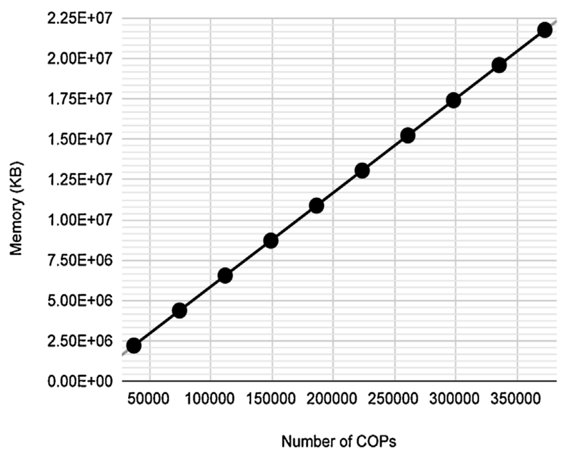
use various of them since each one is optimised for specific functionality. Although FABIOLA is not a new Big Data solution, it aims to be part of the Hadoop ecosystem. FABIOLA provides the necessary components to drive the solution of COPs with distributed data on a Hadoop-based architecture.

In [54], Zhu et al. present various proposals that solve some of the challenges for optimisation problems in distributed information systems. The proposals focus on the optimisation of systems performance and three of them combine optimisation problems in big data environments. First, Zeng et al. [50] emphasised minimising the cost of renting cloud resources for executing MapReduce applications in public clouds, and propose a greedy-based scheduling algorithm to tackle this issue. Secondly, Zhou et al. [53] efficiently distribute data on different devices by considering the asymmetric characteristics among devices and data. To do that, they propose a preference model to quantitatively weight the storage performance imbalance when data are distributed on different devices. Finally, Zhao et al. [52] propose an algorithm to solve the data allocations under multiple constraints in polynomial time. However, these proposals are focused on performance optimisation and not on optimising the results obtained. On the other hand, Chen et al. [8] present a survey where challenges and opportunities of Big Data in data-intensive systems are identified. The authors identified the opportunity to improve decision-making focused on the customer, for instance, by developing customised products. In order to do that, the authors identified the techniques and technologies of Big Data. The authors highlight the need for the application of optimisation techniques to efficiently process a large volume of data within limited run times. Several optimisation techniques are mentioned (cf., Mathematical tools) however they missed out CP as an optimisation technique.

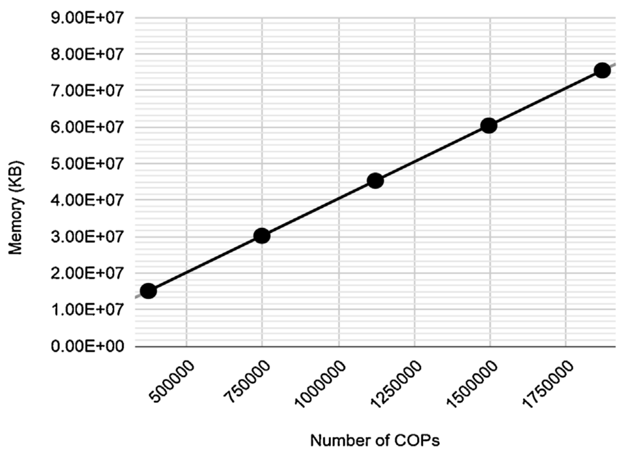
CP presents a challenge in the scalability by solving some hard problems. However, CP has been successfully applied in different domains for solving optimisation problems, such as scheduling and planning. Although there exist several CP tools, such as IBM CPLEX Optimisation [23] and ChocoSolver [37] and solutions that create Constraint Problems according to the constraints extracted from databases [5,18,19,17,38], none of them provides an integrated solution for a Big Data solution. Big Data provides to CP a new perspective concerning the size and volume of data, and it is an excellent opportunity to exploit its possibilities to gain efficiency and optimisation in operational processes [34]. Additionally, Big Data tackles new challenges [16] dealing with automation of decision-making that involves several (millions) decision variables



(a) Memory consumption for DS1.
($y = 22.5x - 1.45 \times 10^6$)



(b) Memory consumption for DS2.
($y = 58.3x + 33327$)



(c) Memory consumption for DS3.
($y = 40.4x + 4898$).

Fig. 17. Memory consumption as the number of COPs scale progressively. Each point represents the average of the total of five executions (Kbytes).

in optimisation of resource consumption, sustainability services, and finance. Nevertheless, the optimisation problems in CP need more flexibility and adaptability, since the exploration of heterogeneous, enormous and dynamic generation of data requires a quick adaptation of optimisation problem to more holistic solutions.

DCOPs have been widely tackled in the literature [15,26,29,48,27,47]. Due to the complexity of the problems (NP-complete, [7,4]) most of the efforts have been done in a theoretical way with the proposition of models, algorithms, and strategies to solve this type of problems. Only ad-hoc applications [29], heuristic approaches [14,13] or simulators [15] have been developed for the community to solve those type of problems. The main challenges in the DCOP solvers are based on overcoming the massive use of memory needed to run the algorithms and to manage the highest number of messages exchanged between nodes. Currently, none approach of DCOPs applied to Big Data has been identified in the literature.

To the best of our knowledge, FABIOLA is the first approach that applied Big Data in CP, although various have been the examples where the necessity to optimise problems with Big Data has been detected [51,24]. There are seminal works related to the application of Big Data in CP. Sunny-CP [1,2] is a portfolio of CP-solvers which can exploit the multicore capabilities of the machines to solve CSPs and COPs. However, this approach lacks applicability in the context of Big Data with multiple data sources with a large-scale amount of data. Cao et al. [6] study how to improve the solution of a large-scale Integer Linear Programming (ILP) problems employing Big Data architecture. ILP is a discrete approach compared with CP. The authors applied ILP to solve traffic flow management. In [28], the authors propose the application of a CP-based scheduler for a Big Data architecture. On the other hand, there is an initiative to create a new language to adapt CP languages for Big Data applications [40], it is currently a very undeveloped approach with no continued development.

Other approaches to Big Data and Optimisation techniques applied to real cases have been studied in [12]. For instance, the maritime logistics problem is modeled as a constraint problem and optimisation techniques have been applied although most of the approaches are focused on machine learning or logic programming techniques.

8. Conclusions and future work

Optimisation problems are found in several real-life scenarios. These optimisation problems become an extraordinary problem when the data involved are in a Big Data environment, which implies a massive quantity of information, that is also distributed and heterogeneous. FABIOLA has been developed to support the definition and resolution of COPs within Big Data scenarios. FABIOLA isolates the resolution of optimisation problems from where the data are and how the optimal outputs are found. Thanks to this isolation, the resolution of optimisation problems in Big Data environments has never been so simple as with FABIOLA. Internally, FABIOLA has been provided with a set of operators to query the optimal information obtained. In order to reduce the computational time and resources needed to solve the optimisation problems, the operators are applied to enrich the capabilities of the resolution of COPs by using the MapReduce operations. Besides, our solution facilitates the COPs modelling through a user interface, FABIOLA-UI, that guides the final users through the application. FABIOLA has been evaluated in real scenarios where the necessity of improving the billing of thousands of customers by using COPs is essential. FABIOLA demonstrated to be a scalable and adaptable solution that improves trivial and stand-alone solutions. However, the most important results are the incredible improvement regarding the performance of execution and resource consumption in optimisation problems. The resolution of optimisation problems in Big Data environments has never been as simple to model and fast to solve as with FABIOLA.

As future work, it would be interesting to extend FABIOLA to support the adaptation of various constraint solvers in the architecture. On the other hand, FABIOLA can be extended in many directions, for instance, related to the inclusion of new operators to optimise the queries.

The possibilities of extends FABIOLA in many directions related to the operators and other scenarios such as distributed optimisation problems are desirable.

Authors' contribution

All the authors are responsible for the concept of the paper, the results presented and the writing. All the authors have approved the final content of the manuscript. No potential conflict of interest was reported by the authors.

Conflict of interest

All the authors are responsible for the concept of the paper, the results presented and the writing. All the authors have approved the final content of the manuscript. No potential conflict of interest was reported by the authors.

Acknowledgements

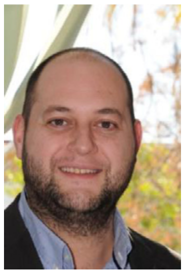
This research was partially supported by Ministry of Science and Technology of Spain with projects ECLIPSE (RTI2018-094283-B-C33) and by Junta de Andalucía with METAMORFOSIS projects; the European Regional Development Fund (ERDF/FEDER); and by the University of Seville with *VI Plan Propio de Investigación y Transferencia* (VI PPIT-US).

References

[1] R. Amadini, M. Gabbrielli, J. Mauro, A multicore tool for constraint solving, Buenos Aires, Argentina, July 25–31, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, 2015, 2015, pp. 232–238.
[2] R. Amadini, M. Gabbrielli, J. Mauro, SUNNY-CP and the minizinc challenge, TPLP 18 (2018) 81–96.
[3] Apache, Apache hadoop, 2018, Online. http://wiki.apache.org/hadoop (Accessed 18 May 2018).
[4] C. Bessiere, I. Brito, P. Gutierrez, P. Meseguer, Global constraints in distributed constraint satisfaction and optimization, Comput. J. 6 (2013).
[5] D. Borrego, R. Eshuis, M.T. Gómez-López, R. M. Gasca, Diagnosing correctness of semantic workflow models, Data Knowl. Eng. 87 (2013) 167–184, http://dx.doi.org/10.1016/j.datak.2013.04.008.
[6] Y. Cao, D. Sun, Large-Scale and Big Optimization Based on Hadoop, Springer International Publishing, Cham, 2016, pp. 375–389.
[7] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence - vol. 1, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1991, pp. 331–337 http://dl.acm.org/citation.cfm?id=1631171.1631221.
[8] C.P. Chen, C.Y. Zhang, Data-intensive applications, challenges, techniques and technologies: a survey on big data, Inform. Sci. 275 (2014) 314–347, http://dx.doi.org/10.1016/j.ins.2014.01.015.
[9] G. Cornuejols, M. Trick, Quantitative Methods for the Management Sciences 45-760. Course Notes, 1998.
[10] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (2008) 107–113, http://dx.doi.org/10.1145/1327452.1327492.
[11] K.R. Duffy, C. Bordenave, D.J. Leith, Decentralized constraint satisfaction, IEEE/ACM Trans. Netw. 21 (2013) 1298–1308, http://dx.doi.org/10.1109/TNET.2012.2222923.
[12] A. Emrouznejad, Big Data Optimization: Recent Developments and Challenges, Springer, 2016.
[13] D. Fernández-Cerero, A. Fernández-Montes, J.A. Ortega, Energy policies for data-center monolithic schedulers, Expert Syst. Appl. 110 (2018) 170–181.
[14] D. Fernández-Cerero, Á.J. Varela-Vaca, A. Fernández-Montes, M.T. Gómez-López, J.A. Alvárez-Bermejo, Measuring data-centre workflows complexity through process mining: the Google cluster case, J. Supercomput. (2019), http://dx.doi.org/10.1007/s11227-019-02996-2.
[15] F. Fioretto, E. Pontelli, W. Yeoh, Distributed constraint optimization problems and applications: a survey, J. Artif. Intell. Res. 61 (2018) 623–698, http://dx.doi.org/10.1613/jair.5565.
[16] E.C. Freuder, B. O'Sullivan, Grand challenges for constraint programming, Constraints 19 (2014) 150–162, http://dx.doi.org/10.1007/s10601-013-9155-1.
[17] M.T. Gómez-López, R. Ceballos, R.M. Gasca, C.D. Valle, Developing a labelled object-relational constraint database architecture for the projection operator, Data Knowl. Eng. 68 (2009) 146–172, http://dx.doi.org/10.1016/j.datak.2008.09.002.
[18] M.T. Gómez-López, R.M. Gasca, Using constraint programming in selection operators for constraint databases, Expert Syst. Appl. 41 (2014) 6773–6785, http://dx.doi.org/10.1016/j.eswa.2014.04.047.
[19] M.T. Gómez-López, R.M. Gasca, Object relational constraint databases for GIS, Encyclopedia of GIS (2017) 1449–1457, http://dx.doi.org/10.1007/978-3-319-17885-1_1598.
[20] Red Eléctrica Group, 'red eléctrica espa na', 2018, Available at http://www.ree.es/es.
[21] P. Gutiérrez Faxas, Distributed Constraint Optimization Related With Soft Arc Consistency, Universidad Autónoma de Barcelona. Departamento Ciencias de la Computación, 2013, Ph.D. thesis.
[22] J.M. Hellerstein, J. Heer, S. Kandel, Self-service data preparation: research to practice, IEEE Data Eng. Bull. 41 (2018) 23–34.
[23] IBM, Ibm-ilog Cplex Optimization. https://www.ibm.com/products/ilog-cplex-optimization-studio (Accessed January 2018).
[24] K. Kolomvatsos, C. Anagnostopoulos, S. Hadjiefthymiades, An efficient time optimized scheme for progressive analytics in big data, Big Data Res. 2 (2015) 155–165, http://dx.doi.org/10.1016/j.bdr.2015.02.001.
[25] A. Labrinidis, H.V. Jagadish, Challenges and opportunities with big data, Proc. VLDB Endow. 5 (2012) 2032–2033, http://dx.doi.org/10.14778/2367502.2367572.
[26] T. Le, T.C. Son, E. Pontelli, W. Yeoh, Solving Distributed Constraint Optimization Problems Using Logic Programming, 2017 arXiv:1705.03916.
[27] A.R. Leite, F. Enembreck, J.P.A. Barthas, Distributed constraint optimization problems: review and perspectives, Expert Syst. Appl. 41 (2014) 5139–5157, http://dx.doi.org/10.1016/j.eswa.2014.02.039.
[28] N. Lim, S. Majumdar, P. Ashwood-Smith, A constraint programming based Hadoop scheduler for handling mapreduce jobs with deadlines on clouds, in: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, ACM, New York, NY, USA, 2015, pp. 111–122, http://dx.doi.org/10.1145/2668930.2688058.
[29] D.K. Molzahn, F. Dörfler, H. Sandberg, S.H. Low, S. Chakrabarti, R. Baldick, J. Lavaei, A survey of distributed optimization and control algorithms for electric power systems, IEEE Trans. Smart Grid 8 (2017) 2941–2962.
[30] Mongo, Mongodb, 2018, Available at https://www.mongodb.com/.
[31] T. Nasser, R. Tariq, Big data challenges, Comput. Eng. Inform. Technol. 4 (2015) 1–10, http://dx.doi.org/10.4172/2324-9307.1000133.
[32] C. Olston, B. Reed, A. Silberstein, U. Srivastava, Automatic optimization of parallel dataflow programs, USENIX 2008 Annual Technical Conference, USENIX Association, Berkeley, CA, USA (2008) 267–273 http://dl.acm.org/citation.cfm?id=1404014.1404035.

[33] OMG, The OMG Data-Distribution Service for Real-Time Systems, 2018, Online. http://portals.omg.org/dds/ (Accessed 18 January 2018).
[34] B. O'Sullivan, Opportunities and challenges for constraint programming, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI Press, 2012, pp. 2148–2152 http://dl.acm.org/citation.cfm?id=2900929.2901033.
[35] L. Parody, Á.J. Varela-Vaca, M.T. Gómez-López, R.M. Gasca, FABIOLA: Defining the Components for Constraint Optimization Problems in Big Data environment, Information System Development - Improving Enterprise Communication, [Proceedings of the 26th International Conference on Information Systems Development, ISD 2017, Larnaca, Cyprus] (2017).
[36] L. Parody, Á.J. Varela Vaca, M.T. Gómez López, R. M. Gasca, Fabiola: Towards the resolution of constraint optimization problems in big data environment, in: Paspallis, N, Raspopoulos M, Barry C, Lang M, Linger H, C. Schneider (Eds.), Advances in Information Systems Development, Springer International Publishing, Cham, 2018, pp. 113–127.
[37] C. Prud'homme, J.G. Fages, X. Lorca, Choco Documentation. TASC – LS2N CNRS UMR 6241, COSLING S.A.S., 2017 http://www.choco-solver.org.
[38] P.Z. Revesz, Introduction to Constraint Databases. Texts in Computer Science, Springer, 2002, http://dx.doi.org/10.1007/b97430.
[39] F. Rossi, P. van Beek, T. Walsh, Handbook of Constraint Programming, Elsevier, 2006.
[40] F. Rossi, V. Saraswat, Constraint Programming Languages for Big Data Applications, 2010 https://github.com/saraswat/C10.
[41] R. Sahal, M.H. Khafagy, F.A. Omara, Exploiting coarse-grained reused-based opportunities in big data multi-query optimization, J. Comput. Sci. 26 (2018) 432–452, http://dx.doi.org/10.1016/j.jocs.2017.05.023.
[42] S. Sai, S. Esakkirajan, Fundamentals of Relational Database Management Systems, vol. 47, 2007, http://dx.doi.org/10.1007/978-3-540-48399-1.
[43] M. Soualhia, F. Khomh, S. Tahar, Task scheduling in big data platforms: a systematic literature review, J. Syst. Softw. 134 (2017) 170–189, http://dx.doi.org/10.1016/j.jss.2017.09.001.
[44] J. Stefanowski, K. Krawiec, R. Wrembel, Exploring complex and big data, Int. J. Appl. Math. Comput. Sci. 27 (2017) 669–679.
[45] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, Hive: a warehousing solution over a map-reduce framework, Proc. VLDB Endow. 2 (2009) 1626–1629, http://dx.doi.org/10.14778/1687553.1687609.
[46] A. Valencia-Parra, A. Varela-Vaca, M. Gómez-López, P. Ceravolo, Chamaleon: framework to improve data wrangling with complex data, ICIS (2019) 205–207.
[47] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, K.H. Johansson, A survey of distributed optimization, Annu. Rev. Control 47 (2019) 278–305, http://dx.doi.org/10.1016/j.arcontrol.2019.05.006.
[48] W. Yeoh, A. Felner, S. Koenig, Bnb-Adopt: An Asynchronous Branch-and-Bound DCOP Algorithm, 2014 arXiv:1401.3490.
[49] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, Apache spark: a unified engine for big data processing, Commun. ACM 59 (2016) 56–65, http://dx.doi.org/10.1145/2934664.
[50] X. Zeng, S.K. Garg, Z. Wen, P. Strazdins, A.Y. Zomaya, R. Ranjan, Cost efficient scheduling of mapreduce applications on public clouds, J. Comput. Sci. 26 (2018) 375–388, http://dx.doi.org/10.1016/j.jocs.2017.07.017.
[51] Z. Zhang, K.R. Choo, B.B. Gupta, The convergence of new computing paradigms and big data analytics methodologies for online social networks, J. Comput. Science 26 (2018) 453–455, http://dx.doi.org/10.1016/j.jocs.2018.04.007.
[52] H. Zhao, M. Qiu, M. Chen, K. Gai, Cost-aware optimal data allocations for multiple dimensional heterogeneous memories using dynamic programming in big data, J. Comput. Sci. 26 (2018) 402–408, http://dx.doi.org/10.1016/j.jocs.2016.06.002.
[53] W. Zhou, D. Feng, Z. Tan, Y. Zheng, Improving big data storage performance in hybrid environment, J. Comput. Sci. 26 (2018) 409–418, http://dx.doi.org/10.1016/j.jocs.2017.01.003.
[54] X. Zhu, L.T. Yang, H. Jiang, P. Thulasiraman, B.D. Martino, Optimization in distributed information systems, J. Comput. Sci. 26 (2018) 305–306, http://dx.doi.org/10.1016/j.jocs.2018.04.020.

Álvaro Valencia-Parra is a computer science Ph.D. student at the University of Seville. His research areas include the improvement of data transformation and data quality fields in Big Data paradigms. His goal is to improve the way in which final users deal with data preparation and specific scenarios in which configuring a Big Data pipeline might be tricky. For this purpose, he is working in the improvement of the data transformation processes by designing Domain-Specific Languages for very specific purposes, user interfaces, and semi-automatic approaches in order to assist users in these tasks.



Angel J. Varela-Vaca received the B.S. degree in Computer Engineering at the University of Seville (Spain) and graduated in July 2008. M.Sc. on Software Engineering and Technology (2009) and obtained his PhD with honors at the University of Seville (2013). Angel is currently working as Assistant Professor at Languages and System Informatics Department at the Universidad Sevilla and belongs to the Idea Research Group. Angel has and leaded various private projects and participated in several public research projects and he has published several impact papers. He was nominated as a member of Program Committees such as ISD 2016, BPM Workshops 2017, SIMPDA 2018. He has been reviewer for international journals such as Journal of Supercomputing, International Journal of Management Science and Engineering Management Multimedia Tools and Applications, Human-Centric Computational and Information Sciences, Mathematical Methods in Applied Sciences among others.

Luisa Parody studied computer engineering (including a minor in systems engineering) at the Universidad de Sevilla (Spain) and graduated with honours in July 2009. She then earned an M.Sc. degree in software engineering and technology (2010) and obtained her international PhD with honours at the Universidad Sevilla (2014). Since 2018, she has been working as an associate professor in Dto. Método Cuantitativos at the Universidad Loyola. She belongs to the IDEA Research Group and has participated in several private and public research projects and has published several high-impact papers.

María Teresa Gómez-López is a Lecturer at the University of Seville and the head of the IDEA Research Group. Her research areas include Business Processes and Data management, and the things that disturb his thoughts are how to improve the business process models including better decisions and enriching the model with Data Perspectives. She has led several private and public research projects and has published several impact papers, among others in Information and Software Technology, Information Systems, Information & Software Technology, or Data & Knowledge Engineering. She was nominated as a member of Program Committees, such as ER 2018, BPM 2017, ISD 2017, ICIQ 2016, PHM 2016, IAwDQ 2013 or CLEIS 2012. She has been reviewing for international journals, such as International Journal of Data and Information Quality, Journal of Systems and Software, Artificial Intelligence in Medicine; Business & Information Systems Engineering Journal or Information Science. She has given keynotes or was invited speaker at the IV Workshop on Data & Artifact Centric BPM in Innsbruck, 5th International Workshop on Decision Mining & Modeling for Business Processes BPM in Barcelona, the X National Conference of BPM in Madrid, in the 28th IBIMA Conference, and in the biannual International Summer School on Fault Diagnosis of Complex Systems.