Partitioning semantics for entity resolution and link repairs in bibliographic knowledge bases

Léa Guizol

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par l'**Université Montpellier II**

Préparée au sein de l'école doctorale **I2S***
Et de l'unité de recherche **UMR 5506**

Spécialité: **Informatique**

Présentée par **Madame Léa Guizol**

Partitioning semantics for entity resolution and link repairs in bibliographic knowledge bases

Soutenue le 21/11/2014 devant le jury composé de :

Mme Nathalie Aussenac	Dr	CNRS, IRIT, Toulouse	Rapporteur
M. Englebert Mephu Nguifo	Pr	Univ. Clermont-Ferrand	Rapporteur
M. Mathieu Roche	Dr	CIRAD, Montpellier	Invité
Mme Marie-Laure Mugnier	Pr	Univ. Montpellier II	Directeur de thèse
Mme Madalina Croitoru	Mdc	Univ. Montpellier II	Co-encadrant

UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

* **I2S:** École doctorale Information Structures Systèmes

Abstract

We propose a qualitative entity resolution approach to repair links in a bibliographic knowledge base. Our research question is: "**How to detect and repair erroneous links in a bibliographic knowledge base using qualitative methods?**" The proposed approach is decomposed into two major parts. The first contribution consists in a partitioning semantics using symbolic criteria used in order to detect erroneous links. The second one consists in a repair algorithm restoring link quality. We implemented our approach and proposed qualitative and quantitative evaluation for the partitioning semantics as well as proving certain properties for the repair algorithms. Portions of this work have been published previously in:

- "Léa Guizol", "Madalina Croitoru". ***Investigating the quality of a bibliographic knowledge base using partitioning semantics***. In FUZZ-IEEE 2014: The annual IEEE International Conference on Fuzzy Systems, pages 948-955.

- "Léa Guizol", "Olivier Rousseaux", "Madalina Croitoru", "Yann Nicolas", "Aline Le Provost". ***An analysis of the Sudoc bibliographic knowledge base from a link validity viewpoint***. In IPMU2014: 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pages 204-213. Springer

- "Léa Guizol". ***Agrégation pour la réparation de liens***. In IC2014: 25es Journées francophones d'Ing. des Connaissances, Clermont-Ferrand, France, 2014.

- "Léa Guizol", "Madalina Croitoru", "Michel Leclère". ***Aggregation Semantics for Link Validity***. In SGAI2014: Research and Development in Intelligent Systems XXX, pages 359-372. Springer International Publishing.

- "Madalina Croitoru", "Léa Guizol", "Michel Leclère". ***On Link Validity in Bibliographic Knowledge Bases***. In IPMU2012: 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, volume Advances on Computational Intelligence, pages 380-389. Springer.

Remerciements

Je souhaite remercier les personnes qui m'ont accueillie au sein du LIRMM et de l'IUT de Montpellier. Je souhaite remercier mes anciens professeurs pour m'avoir transmis l'amour de l'informatique, et Madalina Croitoru pour son encadrement tout au long de la thèse. Je souhaite remercier l'ABES pour les échanges enrichissants qui ont eu lieu durant cette collaboration. Finalement, mais non des moindres, je souhaite remercier ma famille et mes amis pour leur gentillesse, leur soutien et leur patience, et en particulier Mickaël et ma mère pour leurs précieuses relectures. Ainsi que Mya et ses ronrons.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem

Since 2001, ABES (French Bibliographic Agency for Higher Education) has been managing Sudoc[1] (University System of Documentation), a French collective catalog containing over 10 million document descriptions (**bibliographic notices**) and around 2.4 million person descriptions (**authority notices**). Most of the time, bibliographic notices describe books but they can also describe other kinds of documents, like music or movies. Authority notices are used to describe persons and entities that have contributed (as the author or the publisher) to a document described in Sudoc. In this thesis, we focus on book descriptions and limit ourselves to person descriptions.

Bibliographic notices have several attributes (like title, ppn or the unique identifier, publication language, publication date...). Persons who have contributed to a bibliographic notice are represented by **links** that point at authority notices representing those persons. Authority notices have several attributes: ppn (their unique identifier), lists of names and surnames, life dates and sources. The **source** attribute refers to the bibliographic notice for which the authority notice was created.

Example 1 (Link)**.** *Sophocles is represented by the authority notice with ppn "027143619". He has contributed as an author to the play named "Antigone". This play is represented by the bibliographic notice with ppn "166858013", so, there is a link labelled "author" between the bibliographic notice "166858013" and the authority notice "027143619".*

When a librarian needs to register a new book in the Sudoc, (s)he creates a bibliographic notice describing this book. For each person who has contributed to the book (named **contributor**), (s)he has to find whether an authority notice describing this person already exists in Sudoc. Several steps are performed:

[1] A public version of Sudoc is available on the website `http://en.abes.fr/Sudoc/The-Sudoc-catalog`.

- (S)he enters the name and surname (the appellation) of the person. A function selects all the authority notices that have an appellation that might refer to the same person.
- For each selected authority notice, (s)he looks at the bibliographic notices linked to it (the bibliography) in order to compare them to the new bibliographic notice. Unfortunately, it is possible to have a lot of selected authority notices, because of homonyms.
- (S)he selects the authority notice that represents the desired contributor and links it to the new bibliographic notice. If no authority notice fits, (s)he creates an authority notice in order to represent this contributor and links it to the new bibliographic notice. In this particular case, the new bibliographic notice becomes the source of the new authority notice.

The link decision of a librarian is based on the bibliography of the selected authority notice. So, the decision to make a link is based on existing links. Unfortunately, there are erroneous links in Sudoc, that entail new erroneous links. An erroneous link is a link between a bibliographic notice and an authority notice that does not represent the correct contributor, but just a person with a compatible appellation.

This implies the following Research Question:

Research Question. *How to detect and repair erroneous links in a bibliographic knowledge database using qualitative methods?*

Let us present how we answer to this Research Question in the next Section 1.2. We will sum up how results are evaluated in Section 1.3 before detailing the thesis structure in Section 1.4.

1.2 Modelisation and positioning

Let us explain how we propose to detect and repair erroneous links. We will start with the detection part (in Section 1.2.1) and end with the repair part of the problem in Section 1.2.2.

1.2.1 Detecting erroneous links

Let us present here under which hypotheses and how we detect and repair erroneous links in a bibliographic knowledge base.

In this thesis, we assume that an authority notice describing a person contains all the appellations that refers to this person. This implies that only authority notices that have a common or compatible appellation (name and surname) may represent a same person (Work Hypothesis 1). Two appellations are compatible if they may designate the same person (as "Guizol, L.", "Guizol, Lea" or "Guizo, Léa").

Work Hypothesis 1 (Compatible appellation of authority notices)**.** *Only authority notices that share a compatible appellation (name and surname of the represented person) can represent a same real-world person.*

We also assume that documents co-authored by a same person share some characteristics: very dissimilar documents do not share contributors (Work Hypothesis 3) but very similar documents (same titles for example) are likely to share some contributor(s) (Work Hypothesis 2). Similarity of documents is evaluated by comparing their characteristics (like title, publication date, publication language and so on). The more documents seem likely to be pieces of a same work, the more they are considered as similar.

Work Hypothesis 2 (Similar documents and contributors)**.** *Persons who possess very compatible appellations (name and surname) and contributed to very similar documents are the same real-world persons.*

Work Hypothesis 3 (Dissimilar documents and their contributors)**.** *Documents that possess only incompatible characteristics (like publication dates several centuries apart) do not look like each other and do not share common contributors.*

Links between bibliographic notices and authority notices may be incorrect. However, an authority notice is usually created to represent a contributor of a specific bibliographic notice, when no existing authority notice already seems to represent it. In this particular case, the bibliographic notice is said to be the source of the authority notice, and we assume that the link is correct (Work Hypothesis 4). This source notion is introduced in 1.1.

Work Hypothesis 4 (Reliability of links)**.** *Let us consider a link between an authority notice denoted Na_i and a bibliographic notice denoted Nb_j.*

We assume that this link may be wrong, unless Na_i was specifically created in order to represent the contributor of the Nb_j. In this case, Nb_j is said to be the ***source*** *of Na_i.*

In order to detect erroneous links, we have to evaluate links' reliability. Even if an existing link between an authority notice denoted Na_i and a bibliographic notice denoted Nb_j is not correct, we assume that the Na_i appellation is compatible to the appellation of one of the Nb_j contributors. Indeed, Na_i has been linked to Nb_j in order to represent one of its contributors, and the link decision is at least based on the appellation. Because of Work Hypotheses 2 and 3, to check if the links are reliable comes down to check if:

- each authority notice is not linked to dissimilar bibliographic notices,
- all similar bibliographic notices having contributors with compatible appellations are linked to the same authority notice.

In order to compare documents from the point of view of one of their contributors, we create one **contextual entity** per link. A contextual entity represents a link. It also corresponds to a document from the point of view of one of its contributors or a person in the context of one of its documents. It contains all the attributes of the bibliographic notice (title, publication date, ect.), the reliable attribute from the authority notice (the appelation of the contributor), and other attributes related to the link, as the list of the "other contributors" of the bibliographic notice (the contributors who are not represented by the contextual entity). Let us see an Example of a contextual entity.

Example 2 (A person in the context of one of his documents (contextual entity))**.** *In Example 1, we saw that Sophocles was represented by the authority notice with ppn "027143619" linked to the bibliographic notice with ppn "166858013" that represents the play entitled "Antigone".*

Let us detail some of the attributes of the contextual entity that represent this link or Sophocles in the context of "Antigone":

- *appellations:*{"Sophocles","Sophokles","Sophocle"}
- *title: "Antigone"*
- *publication date: "2012"*
- *other contributors:* {"Bousquet, *Joseph*", "Vacquelin, *M.*", "Kalnin-Maggiori, *Hélène*"} *(Those other contributors are the translators, and the editor.)*

The contextual entities possess the appellations of the authority notice pointed at by the link they represent. The contextual entities also represent a person in the context of a document. We recall that authority notices cannot represent the same person unless they have compatible appellations (Work Hypothesis 1). This is extended to contextual entities which cannot represent the same person in the contexts of several documents if they do not have at least a compatible appellation (Work Hypothesis 5).

Work Hypothesis 5 (Compatible appellation of contextual entities)**.** *Only contextual entities with compatible appellations can represent a same real-world person in the context of distinct documents.*

Because contextual entities represent a person in the context of a document, contextual entities which correspond to links pointing to the same authority notice are supposed to represent the same person and to be similar. In the same way, contextual entities which correspond to links pointing at distinct authority notices are supposed to represent distinct persons and being dissimilar. If it is not the case, the Work Hypotheses 2 and 3 are false or there are erroneous links.

Consequently, contextual entities that are similar are supposed to represent links pointing at the same authority notice, and contextual entities that are dissimilar are supposed to represent links pointing at distinct authority notices.

We are interested in sets of similar contextual entities in order to detect which links seem erroneous. The problem of detecting similar objects is exactly the entity resolution problem. However, we are also interested in detecting erroneous links and repairing them. Those tasks are link mining tasks. Indeed, the link mining is the "*process of discovering useful patterns or knowledge from data*" [55] in data that contain links, like bibliographic knowledge bases. Entity resolution can also be a link mining task when data contain links [31]. Link mining tasks according to [31] will be discussed later (in Section 2.2.1).

In order to identify contextual entities that look like each other, we use a partitioning method like [4]. A partition on an object set $\mathbb{O}$ is a set of classes ($\mathbb{O}$ subsets) such that each object of $\mathbb{O}$ is in one and only one class. The partitioning method consists in constructing a partition such that objects that are similar are in the same class and objects that are dissimilar are in distinct classes. Similarity is determined by considering one or several criteria. Like [2], we use symbolic criteria. Symbolic criteria give symbolic comparison values as result for two-object comparisons.

Our interest in symbolic criteria over numerical criteria is because the criteria used mirror how a human expert decides whether two contextual entities belong to the same contributor. An expert can say that the *date* criterion indicates that it is not likely, but the *title* criterion indicates it is likely (like "*Antigone*" from "Sophocles" published in "2012" and in "1568"). We do not necessarily wish to aggregate these comparison values, especially when titles are close but not identical.

A partition is evaluated according to a partitioning semantics. [2] and [4] partition a graph such that vertexes represent the objects to be partitioned and edges are labelled by a value that represents if the vertexes represent similar objects or not. [2] uses a set of criteria in order to determine those values and each criterion can consider the *neutral* value, meaning that the criterion consider the objects as being neither similar nor dissimilar. However, the graph to partition cannot have edges labelled *neutral* in [2] and [4]. In contrast, the two partitioning semantics presented in this thesis accept that comparison values between two objects can be just *neutral*, even in the graph to partition. However, like in [2], some values (*always* and *never*) are shared by all criteria and are more significant than other values. For example, when titles are identical, the *title* criterion gives the *always* value, which means that objects represent the same work.

In this work, we only consider the case with no more or less significant criterion than another criterion. Criteria are said independent from each other (Work Hypothesis 6).

Work Hypothesis 6 (Independancy between criteria)**.** *We consider in this thesis that criteria are independent [14] from each other, which means that no criterion is strictly more important than another to decide which partition is best.*

The thesis work is in entity resolution and in link mining domains, as it is represented on Figure 1.1. It uses symbolic criteria and focuses on the particular case of bibliographic domain.

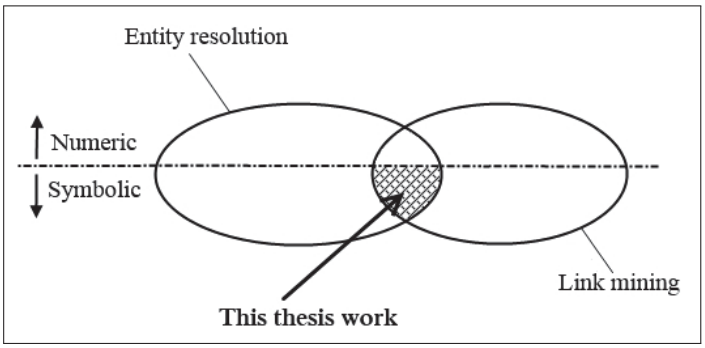

Figure 1.1: Thesis work domain

Evaluating partitions allows us to compare them in order to detect erroneous links. Two partitions are considered in particular:

- the initial partition: the partition such that contextual entities are in the same class if and only if they represent a link pointing to the same authority notice in the Sudoc.
- the human partition: the partition such that contextual entities are in the same class if and only if they represent documents from the point of view of the same real-world person according to a human expert.

In order to detect erroneous links, we expect that the human partition must be a best partition, and we assume that the initial partition is a best partition if and only if it corresponds to the human partition (Work Hypothesis 7). This Work Hypothesis has been evaluated and validated on a Sudoc sample according to librarian experts.

Work Hypothesis 7 (About initial and human partitions)**.** *We suppose that the human partition Ph is a best partition, and that the initial partition Pi is a best partition if and only if Ph is Pi.*

To find erroneous links comes down to find that the initial partition is not a best one. Let us explain the repair of those erroneous links in the following Section.

1.2.2 Repair of erroneous links

The part "repairing erroneous links" takes the partition value v of a best partition on an object set $\mathbb{O}$. If this value is not the value of the initial partition Pi on $\mathbb{O}$, it proposes a repair-partition which has v as partition value, but is as close as possible to Pi. The repair partition is a reorganization of Sudoc links, with the least changes as possible.

1.3 Evaluation

This work has been evaluated for both erroneous link detection and repair link parts. For erroneous links detection:

- we studied how data characteristics (like the criteria's properties or ambiguity in data) influence partitioning algorithms results.
- we have shown that the chosen partitioning algorithms are efficient enough to evaluate almost the entire Sudoc.
- we have shown that the method is relevant in order to detect erroneous links according to ABES experts on a real Sudoc sample.

For link repair, we proved that the proposed algorithms give a repair-partition which has the desired value.

To conclude this Chapter, let us detail the thesis structure in Section 1.4.

1.4 Thesis structure

This thesis contains 6 other Chapters before the conclusion in Chapter 8.

Chapter 2 presents the link mining domain, and focuses on entity resolution. The thesis work is adapted to the particular case of bibliographic domain. This Chapter also shows how a large dataset is divided in small parts to be manageable.

Chapter 3 details the proposed approach for link repair in large bibliographic knowledge databases. This approach is composed of 2 steps: the erroneous link detection step and the link repair step.

Chapter 4 details my point of view of Sudoc context and notions. It also presents criteria that are used in order to compare bibliographic notices.

Chapter 5 presents two partitioning semantics based on symbolic criteria. Those partitioning semantics are used to evaluate partitions in order to compare them and detect erroneous links. This Chapter also discusses the qualitative interest of those partitioning semantics and compare them.

Chapter 6 presents several algorithms that allow us to propose a repair-partition (or reorganization of links) on an object set and according to a partition value. We proved that the repair-partition has the desired partition value and is obtained by a limited number of modifications on the initial partition.

Chapter 7 presents experimentations. Most of them are about the erroneous link detection step, as detailed in Section 1.3. It also presents the future interface that is discussed with ABES experts and will be used in order to show erroneous links to human users, and, in time, to propose repairs to human users.

Chapter 2

Related work

The entity resolution problem is the problem of finding objects that represent the same real-world entity. The entity resolution problem is included in data mining, which is the "*process of discovering useful patterns or knowledge from data*" according to [55]. These data include web pages, databases, pictures... Link mining is a part of data mining, but restraints the process of knowledge discovery into data that contains links, like bibliographic knowledge bases or social networks.

To find objects that represent the same real-world entity when those objects are related to each other by links (like in bibliographic knowledge bases or social networks) is both a link mining problem and an entity resolution problem. In the general case, entity resolution problems often use a distance function or several aggregated criteria in order to compare objects. In this work, we focus on objects which are compared using symbolic values and not numerical values. Indeed, we wish to mirror how a human expert decides whether two books were written by the same person or not. Human experts express themselves with words (like "not so much"), not with numerical values. Finally we adapt these methods in order to detect and repair erroneous links in a bibliographic knowledge base like the Sudoc (a French collective catalog containing document descriptions and person descriptions). This is represented on Figure 1.1 page 20.

The SudocAd project [21] has the aim to predict which person description in Sudoc already represents the author of a new document description. This is used to link a new document description to the person descriptions that represent the authors of the document. As we will see in Section 2.1.2, the SudocAd answers to a link prediction problem (to predict links that will or should exist but do not yet).

Unlike SudocAd, we do not assume that existing links are reliable. On the contrary, we propose a method in order to detect links that should exist but also links that do exist and are erroneous. Detecting erroneous links is out of the scope of the link prediction task.

However, we can represent those links with objects (contextual entities) and compare them in order to detect which links should point at the same person description or at distinct person descriptions but do not. This transforms the problem

of detecting erroneous links into the problem of detecting contextual entities that represent a document from the point of view of a same real-world contributor. This is an entity resolution task.

In this Chapter, we will first start by presenting link mining tasks according to [31] in Section 2.1. Then, we will focus on entity resolution in Section 2.2.

2.1 Link mining tasks related to objects or links

Link mining is a part of data mining, but restraints the process of knowledge discovery into data that contain links, like in bibliographic knowledge bases. [31] distinguishes three families of link-mining tasks:

- **object-related tasks.** Object-related tasks have the aim to find the objects that are similar or closest to a particular object, or the most relevant objects. Object-related tasks include link-based object ranking, link-based object classification, group detection and entity resolution. They will be detailed in the following.
- **link-related task.** Link-related tasks have the aim to predict links that should or will exist between two objects but do not already, and links that should not exist but do. Link-related tasks only include link prediction according to [31]. It will be detailed in Section 2.1.2.
- **graph-related tasks.** Graph-related tasks concern tasks centered around graphs or sub-graphs [1] and not centered around objects, links or object pairs. They namely include the tasks:
  - finding interesting or common sub-graphs (subgraph discovery [51] [44]);
  - classifying a graph as or not as a representation of a concept (graph classification [30] [49]).

In the following Chapters, we will use graphs as the structure containing objects linked together by labelled edges. However, graph-related tasks are not relevant for this thesis because they focus on comparing and detecting entire (sub)graphs but the thesis' objectives focus on links and objects contained in graphs. Indeed, we focus on two research questions:

- **identifying links** that are erroneous and proposing repairs (link-related task). A link connects bibliographic notice (document description) to authority notice (person description) and represents the fact that the person represented by the authority notice is supposed to have contributed to the document represented by the bibliographic notice.

[1] A sub-graph of a graph $G = (O, E)$ is a graph $G' = (O', E')$ such as its object set O' is included into O and its edges set E' is included into E.

- **identifying objects** that are similar (object-related tasks). In our case, to identify objects that are similar means to identify contextual entities that represent documents from the point of view of a same real-world person who has contributed to the document.

This is why we will detail only object and link-related tasks in this Section, before focusing on the most relevant for us, in Section 2.2.

Let us detail link mining tasks related to objects or links. We will start with the entity resolution task (Section 2.1.1) and the link prediction task in Section 2.1.2. Then, we will define and detail the link-based object ranking task (Section 2.1.3), the link-based object classification task (Section 2.1.4), and the group detection task (Section 2.1.5).

2.1.1 Entity resolution

The entity resolution task consists in finding which objects correspond to a same entity without determining *apriori* the considered entities, number of entities or how many objects can refer to the same entities.

This can be interpreted as to identify contextual entities that represent documents from the point of view of a same real-world person that has contributed to the document, which is one of the thesis research questions. The entity resolution task will be further detailed in Section 2.2.

Positioning. Links can be represented by contextual entities, which correspond to the document descriptions from the point of view of one of their contributors. This allows us to adapt the entity resolution task in the aim of detecting contextual entities that represent a document from the point of view of a same real-world person. This problem corresponds to identify links that should point at the same person description but do not and links that should point at distinct person descriptions but do not. Detecting such links comes down to detecting erroneous links, so we use the entity resolution task in order to detect erroneous links.

2.1.2 Link prediction

According to [31], the link prediction problem is the problem of detecting a link that does not exist between two objects. Detecting links is based on links that already exist, and sometimes also on object attributes [64]. For example, [64] predicts the future participation of persons to events, and [54] predicts which persons will soon be co-authors in a new article.

The SudocAd [21] project is a link decision problem adapted to the Sudoc bibliographic knowledge base. It has the aim to help humans to decide which links should be created in order to link a new document description to the already existing person descriptions that represent the authors of the document.

As we said in the beginning of Section 2.2, we distinguish ourselves from SudocAd by not making the hypothesis that existing links are correct. Furthermore, we have the aim to detect erroneous links, which are out of the prediction link scope.

2.1.3 Link-based object ranking

The link-based object ranking task uses the structure of the edges of a graph in the aim of ordering the objects of the graph. This task is adapted to the web and includes namely approaches of HITS [48] and PageRank [66]. Let us detail them.

PageRank [66] orders pages according to the rank of pages that cites them: the more cited a page is by high-ranked pages, the higher its rank. Google has been created in order to show that PageRank is relevant for web pages search. [42] [46] allow PageRank to moderate the importance of a page with respect to the research topic.

HITS [48] classifies the web pages in two types: the authorities (which are often cited) and the hubs (which cite a lot of other pages). It has the aim to find web pages relevant for a given topic (the authorities are more relevant than the hubs). [6] [17] improve HITS by using web pages' content in order to decide how much they are relevant on a given topic. [61] [62] check the stability of both PageRank and HITS when links are modified a bit.

Those approaches are adapted to the web in order to detect most important pages, according to [80]. Several measures exist like centrality degree [29] (the centrality degree of a page is the number of pages linked to it) or as the power centrality [12], which focuses on importance of neighbourhood.

Other approaches are interested in objects linked to the same objects, like [45] or [65]. They evaluate the similarity between two objects based on their shared neighbours. [65] is used to improve the results of entity resolution methods based on record comparisons. For example, an author is seen as linked to his co-authors. If two authors have a close but not identical name and a lot of co-authors in common, they must be the same.

We use the relationships between objects in this thesis work by the intermediary of a criterion (named *otherContributors*) that compares common co-contributors relationships instead of attribute(s) comparison.

2.1.4 Link-based object classification

Like the classic object classification, the link-based object classification [52] [75] [59] has the goal to assign each object to a class already defined. The link-based object classification distinguishes itself from object classification by using classes

of neighbour objects in order to assign a class to an object. The interest of this approach has been remarked by [16] and [63].

Unfortunately, this approach is not adapted to our Sudoc issue because we do not make the hypothesis that we already know all the persons who wrote a document represented in Sudoc: a person can be represented several times or not at all.

2.1.5 Group detection

The group detection task separates a set of objects into object groups (= object classes) such that the number of groups has already been decided and objects in a same group are close to each other but as far as possible from objects in other groups. Two main kinds of approaches are available:

- approaches with a single type of links (links are all similar, with a single type and without attribute).
- approaches based on the stochastic blockmodeling (from the Social Network Analysis domain).

Approaches based on graphs containing only one type of links without attributes are generally based on algorithms aggregating the graph's vertices, or dividing the graph. For example, we predefine the desired number of groups in the *Spectral graph partitioning* [60] [74] [78] approach, which removes the least possible number of edges to get the desired number of groups in the graph. Others [77] use the notion of "edge betweenness" derived from the centrality notion (in Section 2.1.3 [29]): links with a strong "edge betweenness" are progressively removed in order to obtain groups.

Like link based classification (in Section 2.1.4), this task is not adapted to our Sudoc issues because we do not make the hypothesis that we already know the number of persons who wrote a document represented in Sudoc.

The most relevant link mining task for us is the entity resolution task. Indeed, we use entity resolution in order to predict links like SudocAd [21] (which is a link prediction task) and to detect erroneous links. Let us position ourselves in entity resolution approaches in the following Section.

2.2 Entity resolution

As said in Section 2.1.1, the entity resolution task consists in identifying objects that represent the same real-world concept or entity. Those objects can be a bit different because of changed information (a book edited twice with a distinct publication date), missing information, misspeling (as "GUIZOT, Léa" for "GUIZOL, Léa") or incomplete information ("GUIZOL, L." as appellation or "19XX" as publication date).

On the contrary, some objects can seem similar but do not represent the same real-world concepts (persons being homonyms for example).

The entity resolution [7] task is known under many names (entity resolution [7], duplicate detection [11], coreference resolution [67], reference reconciliation [70], fuzzy match [18], object identification [76], deduplication [71], approximate match [34], entity clustering [28], entity matching [13], identity uncertainty [57] [69], merge/purge [43], hardening problem [22], record matching [47], name disambiguation [73], data interlinking [82]).

In the following, we will present different approaches for solving the entity resolution problem in Section 2.2.1. Some approaches focus on how to divide data in order to apply the chosen approach on a data set of a manageable size without missing a comparison between objects that represent the same entity. Those approaches are detailed in Section 2.2.2.

2.2.1 Approaches for entity resolution

In this section, we will go over several types of approaches for the entity resolution task. Approaches for the entity resolution task often use constraints. A constraint is a logical formula that should imperatively be respected.

Here are some constraints. We consider o_1, o_2, o_3, objects to be compared.

- transitivity : if $o_1 \approx o_2$ and $o_2 \approx o_3$ then $o_1 \approx o_3$. This is used for similarity propagation and canonization.
- exclusivity : if $o_1 \approx o_2$ then $o_1 \not\approx o_3$ (an object can match with only one other object at most). This is used for record linkage.
- functional dependency : if $o_1 \approx o_2$ then $o_3 \approx o_4$ (the matching of two objects can imply the match of two others). This can be used, for example, when co-authorships are used to decide which objects represent a same person. In this case, finding that two objects represent the same person can help to find that two of their co-authors, who have close characteristics such as their name, are in fact also a same person.

Those rules can be negatively derived (e.g : if $o_1 \approx o_2$ and $o_2 \not\approx o_3$ then $o_1 \not\approx o_3$).

Record linkage [81] proposes to fuse databases together without conserving several objects that represent the same real-world entity. Each database is supposed to be clean, i.e. without duplicates (two objects describing the same entity) in it. Consequently, the exclusivity constraint (if two objects o_i and o_j represent the same entity e, then there is not another object o_k that represents this entity e) is important, but can be bypassed [41].

The record linkage approach does not fit our problem because we search for duplicates (contextual entities, which represent documents from the point of view of a single real-world person) into a single database.

Similarity propagation [27] [8]. A graph represents objects, the similarity relations between them and possible object merges already done. Objects that are the closest pairwise are recursively merged, as long as it does not break a constraint, until a predefined threshold is hit. Each time two objects o_i and o_j are merged, the similarity relations involving o_i or o_j are updated.

Canonization is merging objects partially representing the same entity. Canonicalization aims to represent in the most possibly complete way the entity's characteristics. For example, [5] proposes an approach that, considering black box merge and comparison functions for pairs of objects, constructs the canonical forms of the represented entities from a set of objects.

The canonization and the similarity propagation approaches do not fit to our problem because we do not want to aggregate duplicates, which are contextual entities that represent documents from the point of view of a single real-world person in our case.

Non-directed probabilistic approach. The non-directed probabilistic approaches are based on Markov's logic [72] [15]. According to [72], Markov logic simply adds weights to first order logic formulas. The logic formulas are used to decide if two objects represent the same entity. Each formula is associated to a weight (a real number between 1 and 0) which allows to use flexible constraints[2]. The interest of flexible constraints to constraints is that flexible constraints may not be verified. The more weighty the flexible constraint is, the more important it is to satisfy it, but it is not mandatory. The goal is to minimize the sum of the unsatisfied constraints' weights.

Non-directed probabilistic approaches do not fit to our problem because they use numerical weighted constraints. The approach for comparing contextual entities has to reflect how human experts decide whether two contextual entities belong to the same contributor. Experts can give contradictory appropriations according to a criterion or another, and we do not wish to aggregate these comparison values.

The partitioning methods based on entity resolution do not need to predetermine the number of expected entities. A partitioning method allows us to divide an object set $\mathbb{O}$ into subsets so that each object of $\mathbb{O}$ is in one and only one subset. Those subsets are called classes. The classes set is a partition. The objects that are in a same class represent the same entity.

Partitioning methods can generate many singletons (classes that contain only a single object) and classes containing only a few objects.

We can distinguish three distinct approaches to partitioning based on entity resolution: the hierarchical partitioning [9], the closest neighbour-based method [19], and the correlation clustering [4] [1]. Let us talk about correlation clustering.

[2] first-order logical formula + weight

The correlation clustering [4] takes a graph with labelled edges. Some edges are closeness edges (labelled +) and others are farness edges (labelled -). The result is a partition of graph vertexes that minimizes the number of unsatisfied edges. An edge is unsatisfied if it is a closeness edge between vertexes in distinct classes, or a farness edge between vertexes inside a same class.

[2] improves on the approach of [4] by adding hard edges. Hard edges can be closeness edges (labelled *always*) or farness edges (labelled *never*) and have to be always satisfied in the final partition result. The number of unsatisfied smooth edges (labelled - or +) is minimized like in [4]. [2] uses the transitivity constraint for hard edges.

In this work, we will use partitioning semantics based on symbolic values that improves on [4] by also accepting edges labelled *neutral*.

When a dataset is too large to fit in central memory, we call it a large dataset. If that is the case, a lot of approaches presented here do not fit anymore because the proposed algorithms process everything at the same time. In that case, data can be split in reasonably-sized datasets in a way that does not influence the obtained results. Let us see how to do it in the next Section.

2.2.2 Strategies for large datasets

When someone wants to apply an algorithm of entity resolution on a large dataset $\mathbb{O}$, this object set can be too large for the desired algorithm. In this case, we can use blocking or canopy strategies in order to divide $\mathbb{O}$ into subsets which can be treated one by one. Let us detail those strategies.

Blocks The blocking strategy consists in dividing the object set $\mathbb{O}$ into subsets (called blocks) which will be treated separately. Each object of $\mathbb{O}$ is in one and only one block. Once it is done, an algorithm that solves an entity resolution problem is applied on each block. This implies that:

- the blocks can be treated in parallel, independently of each other [43];
- if two objects represent the same entity but are not in the same block, they will not be detected. It is important to assume that each object is in the same block as all objects that can refer to the same entity.

The blocking strategy is often based on a blocking function using a hash key (for example, 3 first letters of the name of a person description). Objects are in the same block if and only if they share the same key value.

Unfortunately, the method can be insufficient in order to obtain blocks of a manageable size (the Sudoc contains 6423 document descriptions that have a contributor named "Dupont"). It is possible to combine keys [58] [26] or to proceed by learning a blocking function[10].

When the used algorithm for entity resolution is sensitive to the links between objects, to decide that two objects represent the same entity in one block can affect decisions in other blocks. The canopy strategy is a solution to this issue.

Canopy strategy A function is used to separate the object set $\mathbb{O}$ into subsets called canopies. Contrary to blocks, canopies can share some objects.

[56] only compares objects that share a same canopy. [68] works under the hypothesis that fusing two objects in a canopy can reveal the relevance of fusing other object couples in the neighbour canopies. Two canopies are neighbours if they share an object or more. In order to answer this problem, the solution is the following:

1. Enumerate all canopies on $\mathbb{O}$. Mark them as "to treat".
2. Select a canopy c which is "to treat", mark it "treated" and treat it.
3. If there are some objects recognized as representing the same entity, fuse them and mark the neighbour canopies as "to treat".
4. Return to step 2 until there are no more canopies "to treat".

Positioning In our approach, the Sudoc is divided into subsets which can overlap, like canopies. To find an erroneous link in a canopy can be helpful to find other erroneous links in neighbour canopies because documents that share a contributor are likely to share another contributor, especially if they have the same name. This will be represented by a criterion in Chapter 4. Adding the process of [68] to our erroneous link detection process will allow us to fully use co-contribution relationships in a future development.

Let us detail the approach in the following Chapter.

Chapter 3

Approach

The Sudoc[1] has been a French collective catalog managed by ABES (French Bibliographic Agency for Higher Education) since 2001. It contains around 10 million document descriptions (called **bibliographic notices**) and person descriptions (called **authority notices**). Bibliographic notices are linked to some authority notices by **links**. Those links represent the fact that a person is supposed to have contributed to a document, as an author or illustrator for example. Sudoc notions will be detailed in Chapter 4.

In this Chapter, we will detail the real-world problem for librarians using Sudoc in Section 3.1. Then we will propose our approach to solve this problem in Section 3.2.

3.1 Issue in Sudoc

Everything starts with a librarian who has a book on his table, and must reference it in the Sudoc database. Indeed, one of the most important tasks for ABES experts is to reference a new book in Sudoc. To this end, the expert has to register its title, number of pages, type of publication domains, language, publication date, and so on, in a new bibliographic notice. This new bibliographic notice represents the physical book that the librarian is registering. (S)he also has to register the persons who participated in the book's creation (namely the **contributors**, which includes authors, illustrators, translators, thesis advisor and so on).

In order to do that, for each contributor, (s)he selects every authority notice (named **candidates**) that has a name and surname (called **appellation**) similar to the book contributor. Unfortunately, there is not much information in authority notices because the librarians' policy is to give minimal information, solely in order to distinguish two authority notices that have the same appellation, and nothing more. So the librarian has to look at the bibliographic notices that are linked to

[1] A public audience version of Sudoc is available on the website `http://en.abes.fr/Sudoc/The-Sudoc-catalog`.

the authority notice candidates (the **bibliography** of candidates) in order to decide whether the book at hands seems to be part of the bibliography of a particular candidate. If it is the case, (s)he links the new bibliographic notice to this candidate and looks at the next unlinked contributor. If there is no good candidate, (s)he creates a new authority notice to represent the contributor.

Let us illustrate this on Figure 3.1.

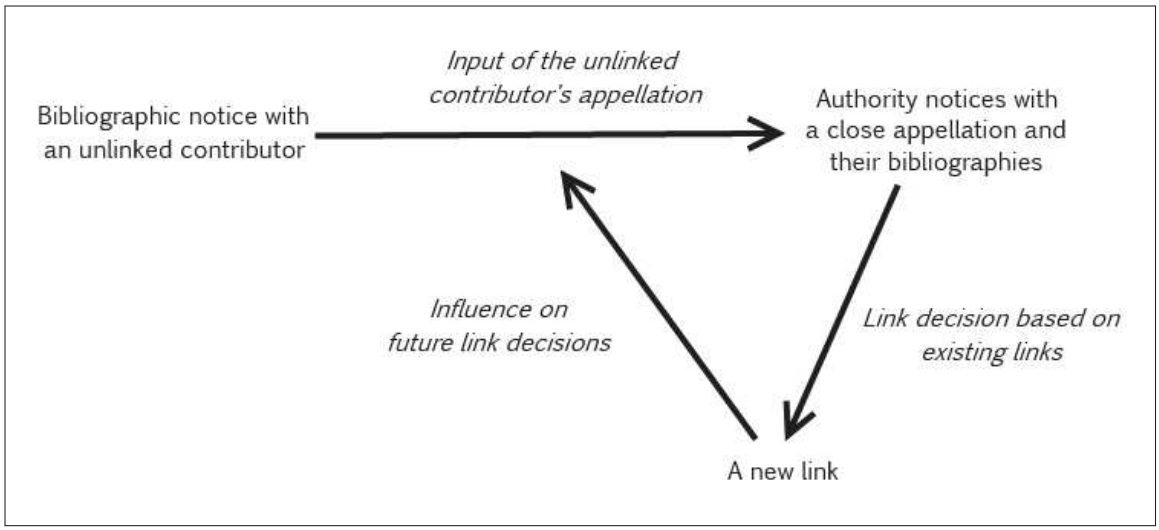

Figure 3.1: Linkage of a new bibliographic notice

This task is fastidious because it is possible to have a lot of candidates for a single contributor (as many as 27 for a contributor named "Bernard, Alain"). The SudocAd project [20] proposes a decision support system to assist librarians to choose authority notices. However, the SudocAd project relies on the hypothesis that the existing data in the Sudoc are clean, and namely:

1. there are no distinct authority notices describing one real-world person,

2. each contributor's name in a bibliographic notice is linked to the "correct" authority notice, and

3. for each bibliographic notice and authority notice, there are no mistakes bigger than misspellings in its attributes' values .

Unfortunately, some persons are represented by several authority notices and there are already link issues in Sudoc. To create new links based on existing erroneous links creates errors, which in turn can create new errors since linking is an incremental process. So we release hypotheses 1 and 2 and focus on erroneous link detection. This will allow us to evaluate the quality of contributor links in Sudoc and improve it. Let us explain our approach in the following Section.

3.2 Approach: to detect and repair erroneous links in Sudoc

In order to help experts to repair erroneous links, we propose a method for the detection and the repair of links between authority notices and bibliographic notices. The Figure 3.2 presents a schema of the method, and indicates which part is detailed in which Chapter. This method is summed up in Figure 3.2 and presented in [24]. This method is a way to answer the Research Question presented page 16: " *How to detect and repair erroneous links in a bibliographic knowledge database using qualitative methods?*". There are four steps represented on Figure 3.2. Let us detail them step by step.

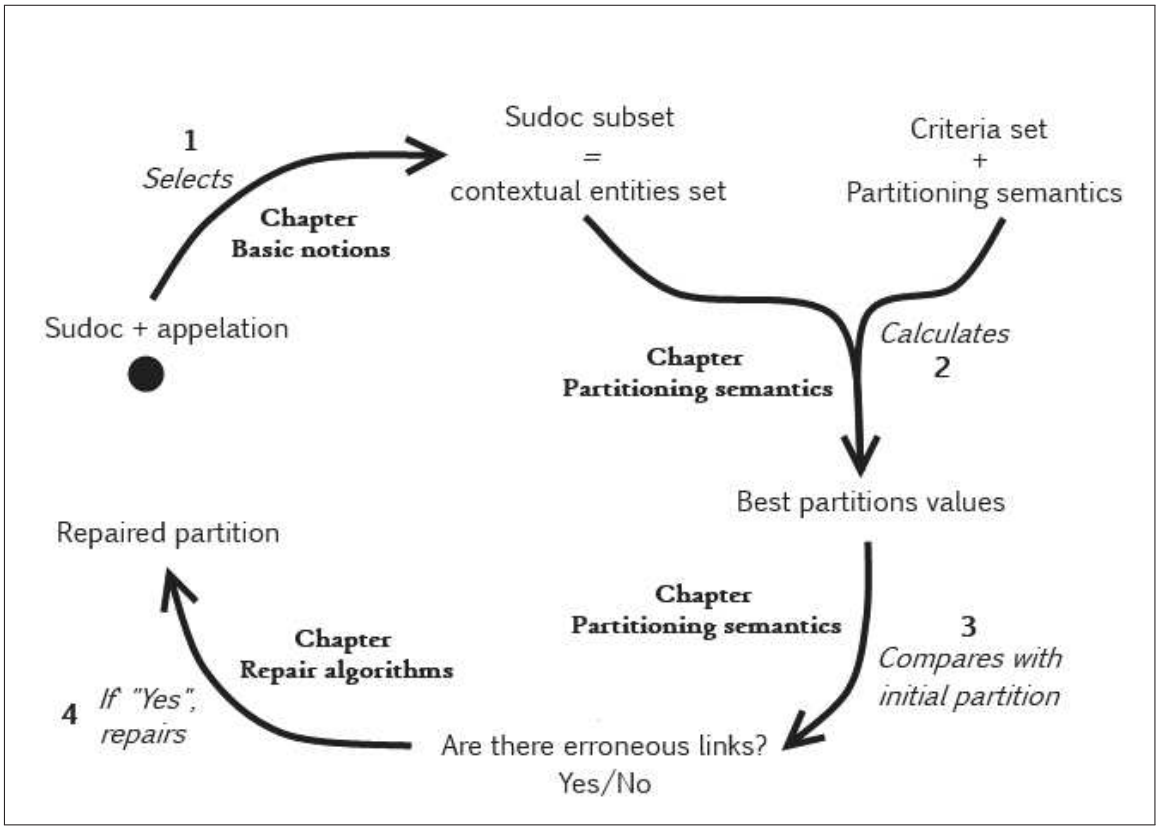

Figure 3.2: General approach

3.2.1 Step 1: finding a Sudoc subset

Step 1 consists in encoding links between authority notices and bibliographic notices as **contextual entities**, which correspond to a document from the point of view of one of its contributors, as it is detailed in Chapter 4 and [24][37]. The appellation of

a contextual entity corresponds to the appellation of the "represented" contributor. Step 1 also selects a Sudoc subset related to an appellation, i.e. a subset of contextual entities that have a similar appellation, because we assume that only contextual entities with a very similar appellation can represent documents from the point of view of the same real-world person (Work Hypothesis 5 page 18). This implies that there is no risk for a bibliographic notice to be linked to an authority notice that has an appellation very dissimilar to the appellation of the true contributor. So there is no need to look for contextual entities with dissimilar appellations that represent the documents from the point of view of a same real-world person.

A Sudoc subset related to an appellation A is the set of contextual entities representing a link between an authority notice that has an appellation close to A and any bibliographic notice (please see Section 4.1.5 page 47 for details). This **step 1** (in Figure 3.2) gives us a set of contextual entities that is denoted $\mathbb{O}$. Each Sudoc subset is a canopy because Sudoc subsets can overlap each other (please see Section 2.2.2 page 30 for further details about canopies).

We assume that the persons who have very similar names and have contributed to very similar documents must be the same real-world person (Work Hypothesis 2 page 17), and, that dissimilar documents must have dissimilar contributors (Work Hypothesis 3 page 17).

Two contextual entities can be compared together with (symbolic) criteria in order to decide whether they represent documents from the point of view of a same real-world person. A symbolic criterion is a function that compares two objects and returns a symbolic comparison value, as detailed in Section 4.2.1 page 48. Criteria are used to decide if compared contextual entities represent a same real-world person. Criteria and contextual entities are related to Sudoc data and are detailed in Chapter 4 and [40].

3.2.2 Step 2: finding the best partitions' values

Contextual entities that are supposed to represent documents from the point of view of a same real-world person are grouped together in order to detect which links are supposed to point to the same authority notice: we create a partition on the considered contextual entity set. A partition on an object set $\mathbb{O}$ is a set of subsets (= classes) of $\mathbb{O}$ such that each object $o \in \mathbb{O}$ is in one and only one class. Each class has to contain at least one object.

According to a partition P, two contextual entities must represent two documents with a common contributor (or two links pointing to the same contributor) if and only if they are in the same class of P.

Unfortunately, criteria are not always sufficient to decide whether two objects are similar, because one can answer "yes" and another "no". Indeed, information given by a specific criterion can be unsuitable for the specifically considered object set.

id	title	date	domain	appellation	ppn
Nc_1	"*Letter to a Christian nation*"		religion	"HARRIS, Sam"	1
Nc_2	"*Surat terbuka untuk bangsa kristen*"	"2008"	religion	"HARRIS, Sam"	1
Nc_3	"*The philosophical basis of theism*"	"1883"	religion	"HARRIS, Sam"	1
Nc_4	"*Building pathology*"	"2001"	building	"HARRIS, Samuel"	2
Nc_5	"*Building pathology*"	"1936"	building	"HARRIS, Samuel"	2
Nc_6	"*Aluminium alloys 2002*"	"2002"	physics	"HARRIS, Samuel"	2

Table 3.1: Example of contextual entities

That is why we propose two partitioning semantics that give a partition value to any partition on $\mathbb{O}$ according to the considered criterion set $\mathbb{C}$. A partitioning semantics evaluates the **partitions** and gives them a partition value. This allows to compare partitions in order to decide which one is the best. Those semantics are detailed in Chapter 5 and presented in [38].

3.2.3 Step 3: detecting erroneous links by comparing partitions

In order to detect erroneous links, we need to define two particular partitions: the human partition and the initial partition on an object set.

The initial partition represents which contributors of which books correspond to a same real-world person according to Sudoc data before erroneous links detection. On the contrary, the human partition represents which contributors of which books correspond to a same real-world person after erroneous links detection according to a human expert.

Definition 1 (The initial partition, denoted Pi)**.** *is the only partition deduced from Sudoc such that two contextual entities Nc_i and Nc_j are in a same class if and only if $Na(Nc_i) = Na(Nc_j)$: they were created to represent links pointing at the same authority notice.*

Example 3 (Initial partition)**.** *Let us represent an object set $\mathbb{O} = \{Nc_1, Nc_2, Nc_3, Nc_4, Nc_5, Nc_6\}$ in Table 1. Each object is a contextual entity, representing a Sudoc link between a bibliographic notice and an authority notice. Id is the contextual entity identity. For each Nc_i of them, the appellation is the $C(Nc_i)$ appellation and the ppn is a way to identify $Na(Nc_i)$.*

$C(Nc_1)$ and $C(Nc_2)$ represent the same person, as $C(Nc_4)$ and $C(Nc_5)$ do. The initial partition on $\mathbb{O}$ is the partition that puts contextual entities together if and only if their C contributor ppn is the same. It is: $Pi = \{\{Nc_1, Nc_2, Nc_3\}, \{Nc_4, Nc_5, Nc_6\}\}$.

Definition 2 (The human partition, denoted Ph)**.** *is the perfect partition according to a human expert: two contextual entities are in a same class if and only if the*

human expert believes that their C contributor corresponds to a unique real person in the real world. If the human partition Ph is not the same partition as Pi, it means that there is a link problem in Sudoc according to the expert who made Ph.

Example 4 (Human partition)**.** *Let us give the human partition, (determined by an expert), on the object set* $\mathbb{O}$ *presented in Example 3 and in Table 3.2.3. This partition is:* $Ph = \{\{Nc_1, Nc_2\}, \{Nc_3\}, \{Nc_4, Nc_5\}, \{Nc_6\}\}$ *because* $C(Nc_1)$ *and* $C(Nc_2)$ *represent a same real-world person, as* $C(Nc_4)$ *and* $C(Nc_5)$ *do.*

Partitioning semantics are used to find the best partition values (**step 2** in Figure 3.2) and to compare partitions together in order to find the most suitable to real-world (**step 3** in Figure 3.2). The partitioning semantics are detailed in Chapter 5 and [38][39]. We make the Work Hypothesis that the human partition Ph is a best partition, and that the initial partition Pi is a best partition if and only if Ph is Pi (Work Hypothesis 7 page 20). This Work Hypothesis allows us to detect erroneous links by comparing the initial partition value to the best partition values.

If the Work Hypothesis 7 is true, detecting link issues in a Sudoc subset comes down to:

- construct contextual entities of the Sudoc subset,
- evaluate the initial partition on those contextual entities (please see the initial partition of a Sudoc subset in Example 3),
- calculate the best partition values on those contextual entities.

The Sudoc subset has a link issue if and only if the initial partition value is not in the best partition values.

In Chapter 7, we used this Work Hypothesis 7 on a real Sudoc sample [40]. We also evaluated scalability of algorithms used to find all best partition values [38] [35]. Finally, we investigated on how criteria and Sudoc data can influence results [36].

3.2.4 Step 4: repairing erroneous links

However, detecting link issues in Sudoc is not enough to help experts to improve the quality of links. That is why we propose in Chapter 6 algorithms that take the initial partition and a best partition value and returns a partition close to the initial one, but with a best value (**step 4** in Figure 3.2). This partition is a proposition of link repairs in Sudoc and is based on the hypothesis that the human partition has a best value (Work Hypothesis 7 page 20).

We presented the global approach of detecting and repairing links in a bibliographic knowledge base. This approach is adapted to the Sudoc bibliographic knowledge base. We will now detail Sudoc notions in the following Chapter.

Chapter 4

Basic notions

The Sudoc is a bibliographic knowledge database. It is used to put together the catalogues of French universities libraries.

The Sudoc contains around 10 million document descriptions (named **bibliographic notices**) and 2 million person descriptions (named **authority notices**). When a person has contributed to a document, there is a **contributor link** between their descriptions. Contributor links are abusively named links in the reminder of this document, unless specified otherwise.

For each contributor link, we create a contextual entity representing it. Those contextual entities are compared to each other to detect erroneous links in order to repair them, as it has been explained in Chapter 3. Criteria used to compare them are defined and detailed in Section 4.2. Let us start by defining and formalising Sudoc notions.

4.1 Formalization of Sudoc

In this Section, we present the formalization of Sudoc notions. Several formalizations of Sudoc are curently available, as explained in Section 4.1.1. We present the way we formalised authority notices (Section 4.1.2), bibliographic notices (Section 4.1.3) and links between them (Section 4.1.4) before defining interesting Sudoc subsets (Section 4.1.5).

4.1.1 Several Sudoc formalizations

In this Section, we will present 3 versions of Sudoc: the original MARC version, the RDF(S) version used by the SudocAd project, and the n-triples version used in this thesis.

MARC version. The Sudoc is initially coded in the MARC language (Sudoc MARC version). Sudoc MARC version is available for non-specialists on the Internet, as we will see in the following Section, but there is less data available when using

the web interface. An example of a bibliographic notice in Sudoc MARC version is shown on Figure 4.1.


001 0192122622@
010##$a0-19-212262-2$d£12.95@
020##aUSb59-12784@
020##aGBbb5920618@
100##$a19590202d1959####|||y0engy0103####ba@
1011#$aeng$cfre@
102##aGBben@
105##$aac######000ay@
2001#$a{NSB} The {NSE} lost domain$fAlain-Fournier$gtranslated from the French by Frank
Davison$gafterword by John Fowles$gillustrated by Ian Beck@
210##$aOxford$cOxford University Press$d1959@
215##$aix,298p,10 leaves of plates$cill, col.port$d23cm@
311##$aTranslation of: Le Grand Meaulnes. Paris : Emile-Paul, 1913@
454#1$1001db140203$150010$a{NSB} Le {NSE} Grand Meaulnes$1700#0$aAlain-Fournier$f1886- 1914$1210##$aParis$cEmile-Paul$d1913@
50010$a{NSB} Le {NSE} Grand Meaulnes$mEnglish@
606##$aFrench fiction$2lc@
676##$a843/.912$v19@
680##$aPQ2611.O85@
700#0$aAlain-Fournier,$f1886-1914@
702#1$aDavison,$bFrank@
801#0aUKbWE/N0A$c19590202$gAACR2@
98700$aNov.1959/209@

Figure 4.1: Representation of the book "*Le Grand Meaulnes*" by "FOURNIER, Alain" in Sudoc MARC version [3]

RDF(S) version. In the context of the SudocAd project [21], the Sudoc data has been translated into an RDF(S) version. RDF is a knowledge representation and reasoning language that is based on triples. Each triple contains an object, a predicate and a subject. The predicate is a property of the object that has the subject as value. It can be represented as an oriented graph such that: the nodes correspond to the objects and the subjects; and the edges are labelled by the predicates. In order to distinguish the object from the subject, edges are oriented from the subject to the object.

The Sudoc RDF(S) version is visible with the Cogui interface [3] as a graph. Let us look at an example of a bibliographic notice visualized with Cogui in Example 5.

Example 5 (A bibliographic notice in RDF(S) version)**.** *Let us consider a bibliographic notice visualized with Cogui on Figure 4.2. In the following paragraphs, letters between parentheses like (A) refer to a node or an edge on Figure 4.2.*

The object `"083396462"` *(A) has a property* `"has_proper_title"` *(B) that has* `"Title : *"` *(C) as value. The object* `"Title:*"` *(C) has the property* `"value"` *(D) which has the value* `"Anglais LP"` *(E). We can deduce that the bibliographic notice of Id "083396462" represents a document that has the title "Anglais LP" because the properties* `"has_proper_title"` *(B) and* `"value"` *(D) link the bibliographic notice to the title of the described document.*

*We can obtain the publication date of a bibliographic notice by following the properties "was_produced_by(2) : *" (F), "has time-span" (H), "is identified by(2)" (J) and "value" (L) in the same way as we did for the title. This allows us to deduce that the bibliographic notice of Id "083396462" represents a document that was published in "1990".*

We can deduce that the bibliographic notice of Id "083396462" represents a document that was published in 1990 and has "Anglais LP" as title, and we need 6 triples (1 per predicate) to extract this piece of information.

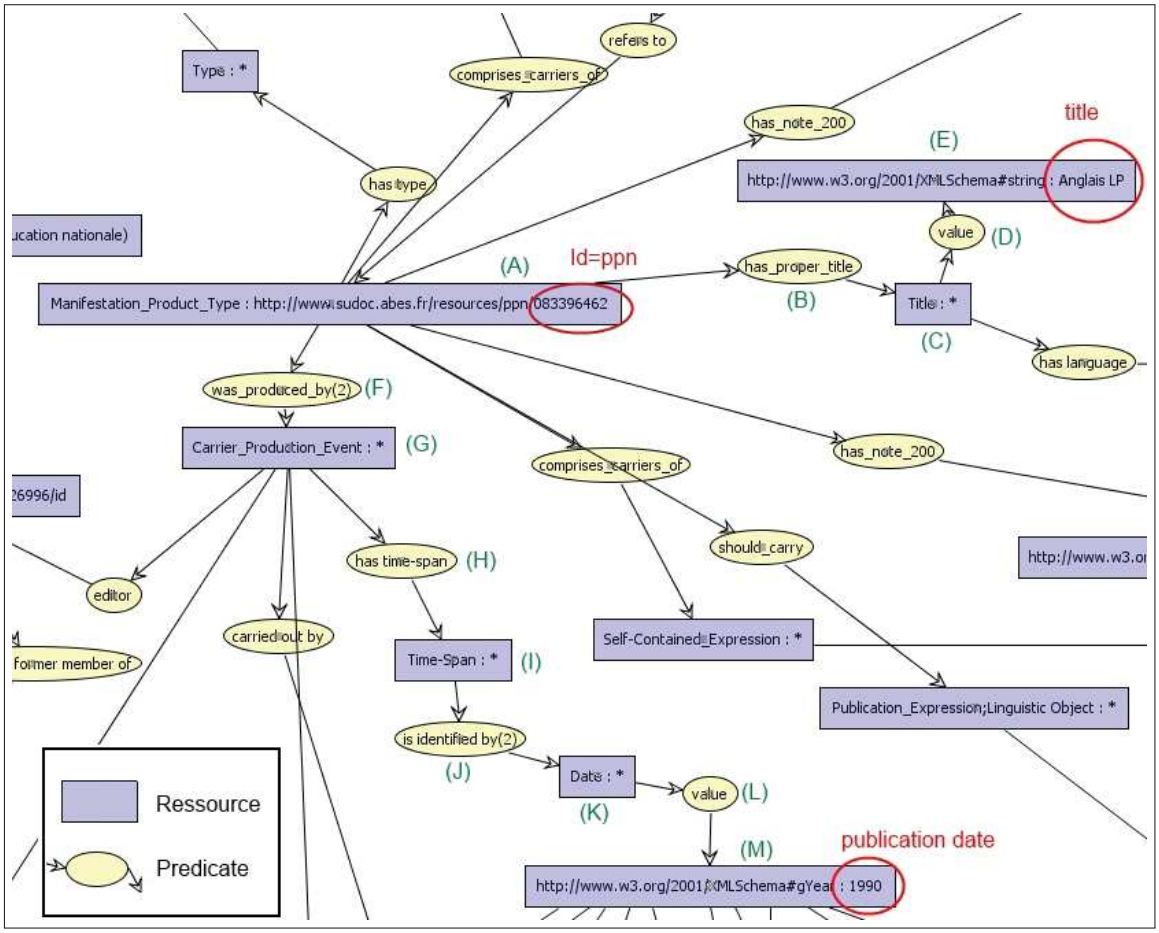

Figure 4.2: RDF version of Sudoc visualized with Cogui

N-triples version. The formalization of Sudoc (in n-triples language, which is also a RDF format) used in this thesis is based on the RDF version. It also uses triples but the data are simplified, as we can see in the following Example 6.

Example 6 (A bibliographic notice in n-triples version)**.** *We saw in Example 5 that 6 triples were required to extract the following information: there is a bibliographic*

notice of Id "083396462" that represents a document published in 1990 under the title "Anglais LP".

It requires only 3 triples in the n-triples version of Sudoc, as shown on Figure 4.3.

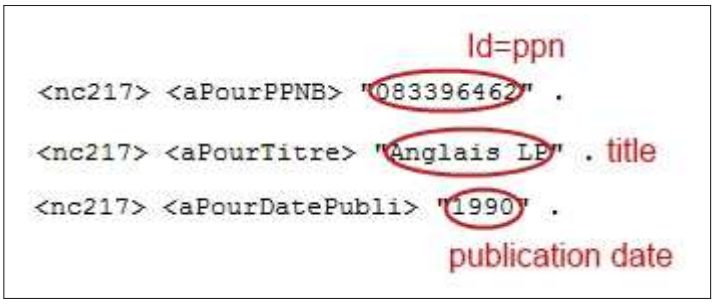

Figure 4.3: n-triples version of Sudoc

Let us detail the formalization of Sudoc notions.

4.1.2 Person descriptions (authority notices)

The descriptions of persons are named **authority notices**, denoted Na, and the j authority notice is denoted Na_j. Authority notices have several attributes:

- the unique Id of the authority notice (the ppn).
- the list of appellations, i.e. names and surnames that can refer to the real-world person.
- sometimes the dates of birth and death.
- the textual descriptions of one or several documents for which the authority notice has been created. The bibliographic notices corresponding to those documents are named the **sources** of the authority notice.

Example 7 (A authority notice)**.** *Let Na_1 be the Sudoc authority notice available on the web in Figure 4.4. This Figure represents the MARC version of Na_1 such as we can see it on the web, as explained in Section 4.1.1.*

We see this Na_1 as the authority notice with the following attributes:

- *ppn: "168603004"*
- *appellations:* {"Leroux, *Alain*"}
- *birth date:*

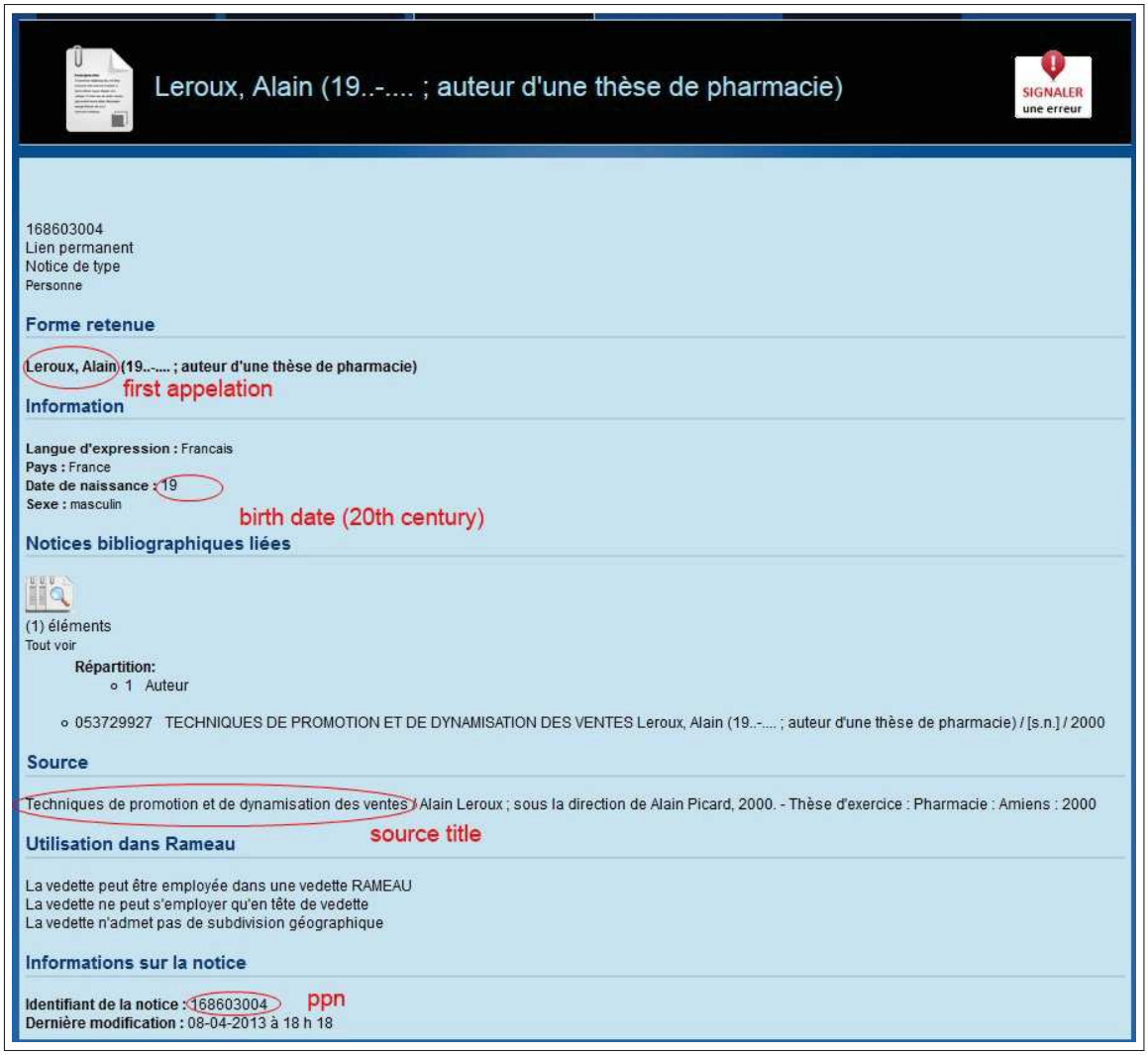

Figure 4.4: An authority notice according to Sudoc web-version

- *death date: none*
- *source: "Techniques de promotion et de dynamisation des ventes / Alain Leroux ; sous la direction de Alain Picard, 2000. - Thèse d'exercice : Pharmacie : Amiens : 2000"*

We can compare it with the web-version of this authority notice shown on Figure 4.4

4.1.3 Document descriptions (bibliographic notices)

The descriptions of documents are called **bibliographic notices**, and are denoted Nb. We use the notation Nb_i to refer to the i bibliographic notice. A bibliographic

notice contains several attributes: the unique identifier of the authority notice (ppn), the title,the publication date, the list of the publication domain codes and the publication language codes. Let us explain these codes before giving a bibliographic notice Example.

The language codes are composed of three letters, this refers to a single language, like "eng" for English language.

The domain codes correspond to the codes of Rameau key-words. Rameau is an indexing language used to annotate the'document descriptions of French libraries[1]. Domain codes are simplified to keep only the domain (as "informatics" instead of "knowledge representation" for example). This simplification can explain the difference between the information about the domains in the Sudoc MARC version and in our Sudoc formalization. In our formalization and RDF(S) version of Sudoc, the domain codes are a string of 3 digits. Let us see an example of two bibliographic notices.

Example 8 (A bibliographic notice)**.** *Let us detail the attributes of two Sudoc bibliographic notices. Let Nb_{11} be the Sudoc bibliographic notice available on the web on Figure 4.5. This Figure represents the MARC version of Nb_{11} such as we can see it on the web, as explained in Section 4.1.1.*

We see this Nb_{11} as the bibliographic notice with the following attributes:

- *ppn: 053729927*
- *title: "Techniques de promotion et de dynamisation des ventes"*
- *publication date: 2000*
- *list of publication domain codes:* {}
- *publication language: "fre" (which is the code for French language).*

We notice that Nb_{11} misses its "list of publication domain codes" attribute. Missing or incomplete attributes are common. Let Nb_{12} be the bibliographic notice with the following attributes:

- *ppn: 041490975*
- *title: "Chan et poésie chez un auteur contemporain : Zhou Mengdie"*
- *publication date: 1993*
- *list of publication domain codes:* {800} *(800 is the domain code that means "litterature")*

[1]The National French Library (BNF) presents this language on the web site: `http://rameau.bnf.fr/` (in French language).

Figure 4.5: A bibliographic notice according to Sudoc web-version

- *publication language: "fre" (which is the code for French language).*

The Example 8 does not show the authors of the documents represented by the bibliographic notices Nb_{11} and Nb_{12}. This piece of information is not available by the attribute values of bibliographic notices but is represented by a **link** between a bibliographic notice and the authority notice that represents the real-world person supposed to have contributed to the creation of the document as an author. We detail it in the next Section.

4.1.4 Links and contextual entities

A **link** represents the fact that the real-world person represented by the authority notice is one of the **contributors**[2] of the document represented by the bibliographic

[2] A contributor of a bibliographic notice is a person who has participated to the creation of the document described by this bibliographic notice.

notice. The contributor links are labelled by the **role** of the contributor: author, illustrator and so on. Let us see the example of a link.

Example 9 (A link)**.** *The person "*Leroux, *Alain" represented by the authority notice Na_1 of Example 7 wrote the thesis "Techniques de promotion et de dynamisation des ventes" represented by the bibliographic notice Nb_{11} of Example 8. This information is represented in Sudoc by a link labelled "author" from Nb_{11} to Na_1. In other words, there is an "author" link between Nb_{11} and Na_1.*

We choose to see Sudoc bibliographic notices from a contributor's view point. A **contextual entity** represents a bibliographic notice from the view point of one of its contributors. Let us consider a particular contributor called the C **contributor**. The contextual entity k is denoted Nc_k. A contextual entity explicitates a contributor link from Sudoc. A contextual entity possesses all the attributes of its bibliographic notice (title, publication language, publication date, publication domain), and five other attributes depending on the C contributor:

- appellations: the appellation list of the authority notice that represents the C contributor.
- role: role of the C contributor.
- other contributors: the list of the authority notices which represent all the contributors of the bibliographic notice but not the C contributor.
- the source label is present if and only if the bibliographic notice is the source of the C contributor. In this case, the contextual entity is a source. Otherwise, the source label is absent.
- the date of birth of the C contributor is present if the contextual entity is a source (i.e. if the source label is present).

We make the hypothesis that a contributor link represented by a contextual entity may be wrong, unless the bibliographic notice is the source of the authority notice (Work Hypothesis 4 page 17). This is why we choose to consider "date of birth" as a contextual entity attribute only if this contextual entity is a source. We denote $sources(Na_i)$ the list of the contextual entities Nc_k that represent a link to Na_i such as Nc_k is a source.

Example 10 (The attributes of a contextual entity)**.** *Let us take the link between the authority notice Na_1 and the bibliographic notice Nb_{11} presented in Examples 7 and 8. There is an "author" link between them as shown in Example 9. We construct the contextual entity Nc_{21} that represents this link. Nc_{21}'s attributes are:*

- *title: "Techniques de promotion et de dynamisation des ventes"*
- *publication date: 2000*

- *list of publication domain codes:* {}
- *publication language: "fre"*
- *appellations:* {"LEROUX, *Alain*"}
- *role: "author".*
- *other contributors:* {("PICARD, *Alain*", *as "thesis advisor"*)}
- *source (because the source of* Na_1 *matches with* Nb_{11}*.)*
- *date of birth: 19??*

Later, we will denote: $cc(Nc_i)$, the C contributor of the contextual entity Nc_i; $Nb(Nc_i)$, the bibliographic notice from which Nc_i was created; and $Na(Nc_i)$, the authority notice from which Nc_i was created and which corresponds to the authority notice that represents $cc(Nc_i)$. The Example 11 illustrates these notations.

Example 11 (An contextual entity)**.** *Let us consider the contextual entity* Nc_{21} *defined in Example 10. This contextual entity was created with the aim of representing the link between the bibliographic notice* Nb_{11} *and the authority notice* Na_1*:*

- $Na(Nc_{21}) = Na_1$ *(*Na_1 *represents* $cc(Nc_{21})$*),*
- $Nb(Nc_{21}) = Nb_{11}$*, and*
- $sources(Na_1) = \{Nc_{21}\}$.

4.1.5 Interesting Sudoc subsets

We explained in the previous Section how we viewed Sudoc data as contextual entities. Let us now explain how we select the relevant Sudoc subsets.

First, we select some contextual entities based on the appellation of their authority notice representing the C contributor. To this aim, an appellation (appellation A) is first chosen when working on contextual entities ("CHRISTIE, A." for example).

All authority notices that could represent a person named A are selected (for example, the authority notices with "CHRISTIE, A.", "CHRISTIE, Agatha" or "CHRISTIE, Adam" in their list of appellations attributes are selected for the appellation "CHRISTIE, A."). This is a service provided by the ABES. This service uses the same function as the one that is used to select authority notices when a librarian wants to link the contributor of a new bibliographic notice to an authority notice (described in Section 3.1 page 33).

Each contributor link between a selected authority notice and any connected bibliographic notice is retrieved. Contextual entities that represent such a link are

selected and form the **Sudoc subset related to** the appellation A (for example, the Sudoc subset related to "CHRISTIE, A.").

Librarians use the appellation attribute of authority notices to select the authority notices that might represent the contributor of a document. We use the appellation attribute in order to select a Sudoc subset that contains bibliographic notices that might share the same real-world person as their C contributor. We make the hypothesis that only the authority notices with a close appellation can represent a same real-world person (Work Hypothesis 1 page 17). As a consequence, only contextual entities with a close appellation can share the same real-world person as their C contributor (Work Hypothesis 5 page 18).

Once obtained, the Sudoc contextual entities are compared with respect to symbolic criteria in order to know if their C contributors might be the same real-world person. The approach will be detailed and contextualised in Chapter 3. We will now detail the criteria that are used in order to determine if two contextual entities share the same real-world person as their C contributor.

4.2 Criteria to detect link issues in Sudoc

As explained in Section 3.2 page 35, contextual entities are compared together in order to group contextual entities that represent a bibliographic notice from the point of view of a same real-world person that contributed to the described documents. Contextual entity groups are made according to a partitioning semantics (detailed in Chapter 5). These partitioning semantics use symbolic criteria in order to compare contextual entities.

Symbolic criteria are formally defined in Section 4.2.1 before presenting the actual criteria used to compare Sudoc contextual entities in Section 4.2.2.

4.2.1 Symbolic criteria

In order to detect erroneous links in Sudoc, we would like to partition a bibliographic notice set with respect to several criteria. In the general case, we would like to partition an object set $\mathbb{O}$ with respect to a criterion set $\mathbb{C} = \{C_1, ..., C_n\}$.

In the general case, a criterion $C_i \in \mathbb{C}$ is a function that gives a **comparison value** for any couple of objects in $\mathbb{O} * \mathbb{O}$.

The comparison value is discrete, in a partially ordered set $V = \{never\} \cup V^C_{far} \cup \{neutral\} \cup V^C_{close} \cup \{always\}$, where:

- *never* and *always* mean that the objects are strictly different (respectively identical).
- *neutral* means that there is not enough information about the objects or the information is meaningless.

- V^C_{far} and V^C_{close} are two totally ordered value sets ($<$) of values giving a farness degree (respectively closeness degree) between objects. We denote $V^C_{far} = \{-, --, ...\}$ (respectively $V^C_{close} = \{+, ++, ...\}$) the sets of farness (respectively closeness) values of the criterion C such that $neutral < - < -- < ... < never$ (respectively $neutral < + < ++ < ... < always$). $a < b$ means that b is stronger or more intense than a: if $a, b \in V^C_{close}$, two objects are more similar according to C if their comparison value is b than if it is a. Respectively, if $a, b \in V^C_{far}$, two objects are more dissimilar according to C if their comparison value is b than if it is a.

never, *always* and *neutral* values are common for all criteria, but V^C_{close} and V^C_{far} are specific to the criteria named C.

Let us formally define the used criteria.

Definition 3 (Symbolic criterion)**.** *Let $\mathbb{O}$ be an object set. A criterion C is a couple (e_C, V_C) such as:*

- *V_C is a set of comparison values: $V_C \supseteq \{never\} \cup V^C_{far} \cup \{neutral\} \cup V^C_{close} \cup \{always\}$ and $V_C \subseteq V^C_{far} \cup V^C_{close}$.*
- *$<$ is totally ordered relation on $\{never\} \cup V^C_{far} \cup \{neutral\}$ and $\{neutral\} \cup V^C_{close} \cup \{always\}$ sets.*
- *e_C is a binary operation, commutative on $\mathbb{O}$, which takes its results in V_C.*

We represent by $\mathcal{C}$ the set of all possible criteria, and $\mathbb{C} \subset \mathcal{C}$ the subset of the criteria of interest.

A criterion can be a **closeness-criterion** (Definition 4), a **farness-criterion** (Definition 5) or both closeness and farness-criterion depending on the comparison values it can give. A closeness-criterion c is a criterion that can give a closeness or *always* comparison value to two objects. A farness-criterion is a criterion that can give a farness or *never* comparison value to two objects. This notion will be important later in Chapter 7.

Definition 4 (Closeness criterion)**.** *A closeness-criterion C is a criterion that can give a closeness or an always comparison value to two objects: $V_C \cap (V^C_{close} \cup \{always\}) \neq \{\}$.*

Definition 5 (Farness criterion)**.** *A farness-criterion is a criterion that can give a farness or a never comparison value to two objects: $V_C \cap (V^C_{far} \cup \{never\}) \neq \{\}$.*

Let us detail the criteria developed and used to compare contextual entities in Sudoc.

4.2.2 Criteria used to compare contextual entities

We developed and implemented nine criteria in the aim of comparing contextual entities together. We use symbolic criteria over numerical criteria in order to compare contextual entities because:

- Criteria model how an expert compares contextual entities. Criteria are based on one or several attributes of contextual entities.
- Experts can explain that, according to an attribute set, contextual entities seem "more or less" to represent documents from the point of view of a same real-world person, but
- We do not know how much the "more" according to an attribute set is more important than the "less" according to another attribute set, except when they are absolutely certain (which is represented by the comparison values *always* and *never*).
- Consequently, we do not wish to aggregate comparison values as we can do with numerical values, so used comparison values are symbolic.

Each of these nine criteria has been discussed with ABES experts. Four of them are derived from the SudocAd project [21] (*appellation*, *date*, *title*, *domain*) that aims to predict a link between a new bibliographic notice and the authority notices already in Sudoc (that represent the new bibliographic notice contributors, as explained page 34). Each criterion will be illustrated by an example comparing the contextual entity Nc_{21} defined in Example 10 and the contextual entity Nb_{22} of the Example 12.

Example 12 (A contextual entity to be compared)**.** *The bibliographic notice Nb_{12} described in Example 8 is linked to the authority notice Na_2 which has 168603160 as ppn. Let us present all the attributes of the contextual entity Nb_{22} that represents this link:*

- *title: "Chan et poésie chez un auteur contemporain : Zhou Mengdie"*
- *publication date: 1993*
- *list of publication domain codes:* {800} *(800 is the domain code which means "litterature")*
- *publication language: "fre" (which is the code for French language).*
- *appellations:* {"Leroux, *Alain*"}
- *role: "author".*

- *other contributors: {(027073920, as "thesis advisor"), ("*Pimpaneau, *Jacques", as "thesis advisor")}*
- *source? "yes" (because the source of Na_2 matches with Nb_{12}.)*
- *date of birth: 19??*

Some of the proposed criteria are farness-criteria (*thesis*, *thesisAdvisor*, *date*, *appellation*, *language*), closeness-criteria (*title*, *otherContributors*), and others are both (*role*, *domain*). Criteria may use one or several attributes in order to compare contextual entities. Each of those criteria gives the *neutral* comparison value when a required attribute of a compared contextual entity is unknown. First we will present farness-criteria, then closeness-criteria and finally criteria that are both farness and closeness criteria. In the following criterion descriptions, only the cases with a not *neutral* comparison values are specified. Let Nc_i, Nc_j be two contextual entities.

Farness-criteria

- The *appellation* criterion is a particular farness-criterion. Indeed, it compares appellation lists to determine which contextual entities cannot have a same C contributor. When it is certain (as when appellations are "Conan Doyle, Arthur" and "Christie, Agatha"), it gives a *never* comparison value, that forbids other criteria to compare the concerned authority notices together. It is based on the comparisons used in the SudocAd project [21] in order to compare family names and surnames. This criterion is distinct from the function that is used to select Sudoc subsets related to an appellation defined in Section 4.1.5.

Example 13 (*appellation* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. The appellation attribute of Nc_{21} is exactly the appellation attribute of Nc_{22}: {"*Leroux, *Alain"}, so $appellation(Nc_{21}, Nc_{22}) = neutral$.*

- The *date* criterion is a farness-criterion. For 100 (respectively 60) years at least between the publication dates, it gives $--$ (respectively $-$) comparison value.

Sometimes, the publication dates are not years but decades or centuries. This is why the publication dates are compared not as numbers but as intervals. For the *date* criterion, the minimal distance between the compared date-intervals is the one that matters.

Example 14 (*date* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13 so we can compare date attributes. The publication date of Nc_{21} is "2000" and the publication date of Nc_{22} is "1993" so there are 7 years at most between the publication dates and $date(Nc_{21}, Nc_{22}) = neutral$.*

• The *language* criterion is a farness-criterion. When publication languages are distinct and none of them is English ("eng" code) or undefined ("und" code), *language* returns $-$ value, and in other cases, *neutral* value.

Example 15 (*language* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13, so we can compare publication language attributes. The publication language of both Nc_{21} and Nc_{22} is the same, "fre" or French so $language(Nc_{21}, Nc_{22}) = neutral$.*

• The *thesisAdvisor* criterion is a farness-criterion. $thesisAdvisor(Nc_i, Nc_j) = --$ means that Nc_i and Nc_j have a same C contributor if their attributes allow to detect that this contributor has supervised a thesis before submitting his/her own thesis. $thesisAdvisor(Nc_i, Nc_j) = -$ means that Nc_i and Nc_j have a same C contributor if their attributes allow to detect that this contributor has supervised a thesis submitted only 2 years after submitting his/her own thesis. This criterion uses several attributes in order to calculate a comparison value: role, other contributor and publication date attributes. A contextual entity represents a thesis from the point of view of a thesis advisor if the role is "thesis advisor". A contextual entity represents a thesis from the point of view of its author if the role is "author" and one of the other contributors has contributed as a "thesis advisor".

Example 16 (*thesisAdvisor* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13, both are theses according to Example 17, but neither of them has thesis advisor as role, so $thesisAdvisor(Nc_{21}, Nc_{22}) = neutral$.*

• The *thesis* criterion is a farness-criterion. $thesis(Nc_i, Nc_j) = -$ means that Nc_i, Nc_j are contextual entities that represent distinct theses (not the same title, recognized thanks to the *title* criterion, that will be presented further) from their "author" point of view. $thesis(Nc_i, Nc_j) = --$ means that Nc_i, Nc_j have also been submitted simultaneously (3 years at most between submissions).

Example 17 (*thesis* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. Since $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13, roles are "author" and both have a thesis advisor as other contributor. Nc_{21} and Nc_{22} both represent a thesis from the point of view of their author. Furthermore, the titles are distinct according to Example 18 ($title(Nc_{21}, Nc_{22}) = neutral$). Publication dates are separated from 7 years, so publication dates are not too close but $thesis(Nc_{21}, Nc_{22}) = -$.*

Closeness-criteria

- The *title* criterion is a closeness-criterion. This criterion can give an *always* value and 3 closeness comparison values. It is based on a Levenshtein comparison[3] [53]. For each comparison of titles, the Levenshein comparison gives a number i between 0 (totally different) and 1 (identical). We discrete this in the following manner:
  - if $i > 0.95$, the comparison value is *always*
  - if $0.95 > i \geq 0.8$, the comparison value is $+++$
  - if $0.8 > i \geq 0.6$, the comparison value is $++$
  - if $0.6 > i \geq 0.5$, the comparison value is $+$

  It is useful to determine which contextual entities represent the same work (eventually edited several times). It is also used by the *thesis* criterion.

Example 18 (*title* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. appellation(Nc_{21}, Nc_{22}) = neutral according to Example 13, so the appellation criterion does not exclude that Nc_{21} and Nc_{22} could have a same C contributor and we can compare title attributes. The title of Nc_{21} is "Techniques de promotion et de dynamisation des ventes" and the title of Nc_{22} is "Chan et poésie chez un auteur contemporain : Zhou Mengdie", so the title$(Nc_{21}, Nc_{22}) = neutral$.*

- The *otherContributors* criterion is a closeness-criterion. It counts the other contributors in common by comparing their authority notices. One other common contributor gives $+$ comparison value. Several other common contributors give $++$ comparison value.

Example 19 (*otherContributors* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. Since appellation(Nc_{21}, Nc_{22}) = neutral according to Example 13, we can compare other contributors attributes. The title of Nc_{21} has 1 other contributor, that is not any of the two other contributors of Nc_{22}, so otherContributors$(Nc_{21}, Nc_{22}) = neutral$.*

Both farness and closeness-criteria

- The *role* criterion returns $+$ when contributor C roles are the same (except for some pairs of roles as "thesis advisor" and "author"), or $-$ when distinct (except for common roles as "author", "publishing editor" or "collaborator").

[3] The Levenstein distance measure corresponds to the minimal number of transformations between two strings. A minimal transformation can replace a character by another, remove or add a character.

Example 20 (*role* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13, so we can compare their role attribute. Role of both Nc_{21} and Nc_{22} are the same, "author", that is a very common role so $role(Nc_{21}, Nc_{22}) = neutral$.*

• The *domain* criterion compares the lists of domain codes. Domain codes are compared pair-wise. $domain(Nc_i, Nc_j)$ compares the lists of domain codes, and gives closeness (respectively farness) comparison values if every Nc_i domain code is close (respectively far) from a Nc_j domain code and the other way around. *neutral* value is given by default. The domain codes are compared pair-wise as "identical", "very close", "close", "not so close", "a bit far", "far", "very far". $domain(Nc_i, Nc_j)$ compares list of domain codes:

- $++++++$ if Nc_i and Nc_j domain code lists are included in each other;
- $+++++$ if every Nc_i domain code is at least "very close" from a Nc_j domain code and the other way around;
- $++++$ if every Nc_i domain code is at least "close" from a Nc_j domain code and the other way around;
- $+++$, $++$ or $+$ if there are some "identical", "very close" or "close" relation between the domain codes of Nc_i and Nc_i, but no "not so close", "a bit far", "far", or "very far" relation between the domain codes of Nc_i and Nc_i;
- *neutral* if there are some "identical", "very close" or "close" and some "not so close", "a bit far", "far", or "very far" relation between the domain codes of Nc_i and Nc_i;
- $-$ if every Nc_i domain code is "not so close", "a bit far", "far", or "very far" from each Nc_j domain code and the other way around;
- $--$ if every Nc_i domain code is "a bit far", "far", or "very far" from each Nc_j domain code and the other way around;
- $---$ if every Nc_i domain code is "far", or "very far" from each Nc_j domain code and the other way around;
- $----$ if every Nc_i domain code is "very far" from each Nc_j domain code and the other way around.

Example 21 (*domain* criterion)**.** *Let us compare the contextual entities Nc_{21} and Nc_{22} defined in Examples 10 and 12. $appellation(Nc_{21}, Nc_{22}) = neutral$ according to Example 13, so we can compare their list of domains attributes. Nc_{22} list of domains is $\{800\}$ but Nc_{21} list of domains is missing so $domain(Nc_{21}, Nc_{22}) = neutral$.*

We presented in this Chapter how we saw Sudoc data and how we compared contextual entities. Let us present the partitioning semantics used in order to detect erroneous links in Sudoc in the following Chapter.

Chapter 5

Partitioning semantics

We present in this chapter two partitioning semantics using symbolic criteria. The input of a partitioning semantics is a **partition** [1] P on a set of objects and a set of criteria such as each criterion gives a comparison value for each pair of objects. The output is the partition value of P according to the criterion set. Partition values allow us to compare partitions on the same object set, and to decide which one is the best partition. The formal definition is given in Section 5.1.1.

Partitioning semantics are used in order to compare contextual entities of Sudoc subsets. This allows us to detect if there are erroneous links in the Sudoc subsets considered by determining if the initial partition is a best partition, as it has been explained in Section 3.2.3 page 37.

Partitioning semantics evaluate partitions according to one or several criteria. As previously explained in Section 4.2.1, a symbolic criterion C is a function that returns a symbolic value for two objects. This symbolic value is discrete and could be *neutral*, *always*, *never*, in V^C_{far} (farness value) or V^C_{close} (closeness value). Each criterion has its own sets of farness and closeness values, as explained in Section 4.2.1. Algorithms used to find best partitions focus on best partition values instead of best partitions because there are Bell number ($\mathbf{B}_n$) possible partitions on an object set. If this object set has a size n, the Bell number is, with $\mathbf{B}_0 = \mathbf{B}_1 = 1$:

$$\mathbf{B}_n = \sum_{k=0}^{n-1} \binom{n-1}{k} \mathbf{B}_k$$

We will start with common notions for both semantics, as the general definition of a partitioning semantics in Section 5.1. Then, we will present the global semantics in Section 5.2 and the local semantics in Section 5.3, which permits to improve global semantics. We will finish by giving some details about implementation in Section 5.4.

[1] A **partition** P of an object set X is a set of **classes**(X subsets) such as each object of X is in one and only one P class.

5.1 Basic notions for both semantics

In this Section, we present important common notions for both global and local semantics. We start with a general definition of partitioning semantics Section 5.1.1. Then, we explain how the problem of partitioning an object set according to a criterion set is viewed as a graph partitioning problem (Section 5.1.2). In Section 5.1.3, we will be able to present common notions about partition values for both global and local semantics.

5.1.1 General partitioning semantic definition

Let us define what a partitioning semantics is in a general case. In the general case, a partitioning semantics gives a value to any partition on an object set. These values serve to compare partitions. Let us formally define what a partition is (Definition 6) before defining a partitioning semantics in Definition 7.

Definition 6 (Partition). *A partition P of an object set $\mathbb{O}$ is a set of classes ($\mathbb{O}$ subsets) such that there are no empty class and each object of $\mathbb{O}$ is in one and only one P class.*

Let o be an object in $\mathbb{O}$, and P a partition on $\mathbb{O}$. The class of P containing o is denoted $class(o, P)$ or abusively $class(o)$ if P is obvious.

Example 22 (Partition). *Let $\mathbb{O}$ be an object set such that $\mathbb{O} = \{a, b, c, d, e\}$. $P = \{\{a, c\}, \{b, d\}, \{e\}\}$ is a partition on $\mathbb{O}$. $class(b, P) = \{b, d\}$.*

Definition 7 (Partitioning semantics). *Let $\mathbb{C}$ be a set of criteria on the object set $\mathbb{O}$, $\mathbb{P}$ the set of possible partitions on $\mathbb{O}$ and $\mathbb{VP}$ the set of possible partition values. A semantics of best partitioning is composed of:*

- *a function that gives a value for a partition: $v : \mathbb{P}, \mathbb{C} \mapsto \mathbb{VP}$, and*
- *an order on $\mathbb{VP}$.*

*Let P be a partition on an object set $\mathbb{O}$. We denote $v(P, C)$ (respectively $v(P, \mathbb{C})$ the **partition value** of P with respect to the criterion C (respectively the criterion set $\mathbb{C}$). If C (respectively $\mathbb{C}$) is obvious, we denote abusively $v(P)$ the value of P with respect to C (respectively $\mathbb{C}$).*

Let P_i, P_j be two partitions on the same object set and vp_i, vp_j their respective partition values according to a partitioning semantics and a criterion set. Comparing P_i and P_j, comes down to comparing their values. We denote:

- *"vp_i equal vp_j": $vp_i = vp_j$ (P_i and P_j are said equivalent, denoted $vp_i \approx vp_j$);*
- *"vp_i is strictly preferred to vp_j": $vp_i \succ vp_j$;*
- *"vp_i is preferred to vp_j": $vp_i \succeq vp_j$;*

- *"vp_i is not comparable to vp_j": $vp_i \diamond vp_j$ (means that $vp_i \succeq vp_j$ and $vp_i \preceq vp_j$ are both false).*

$P_i = P_j$ means that P_i is strictly the same partition as P_j. We abusively use the same symbols ($\succ$, $\succeq$, $\diamond$) and vocabulary to compare partitions and to compare their values.

In order to compare two partitions P_i, P_j on the same object set $\mathbb{O}$ with respect to a criterion set $\mathbb{C}$, we compare P_i and P_j values returned by the semantics considered with respect to $\mathbb{C}$. Two partitions are equivalent ($P_i \approx P_j$) when they have the same value ($v(P_i) = v(P_j)$). A partition is strictly preferred to another one ($P_i \succ P_j$) if and only if its value is preferred ($v(P_i) \succ v(P_j)$). A partition is preferred to another one ($P_i \succeq P_j$) if and only if its value is preferred ($v(P_i) \succeq v(P_j)$). Partitions are not comparable to each other ($P_i \diamond P_j$) when their values are also not comparable to each other ($v(P_i) \diamond v(P_j)$).

We presented the general definition of a partitioning semantics and the notations for comparing partitions and partition values. Let us transform the problem of partitioning an object set into a graph partitioning problem.

5.1.2 Graph representation of a partitionning problem

In this section, we represent the problem of partitioning an object set by a graph partitioning problem.

For each criterion $C \in \mathcal{C}$ and a set of objects $\mathbb{O}$, we define a criterion graph. This graph represents the objects of $\mathbb{O}$ by vertexes and the C comparison values by labels of edges.

Definition 8 (Criterion graph)**.** *We represent a criterion C on an object set $\mathbb{O}$ by a criterion graph $G_C = (S, E_C)$ such that vertexes are the elements contained in $\mathbb{O}$ and edges are labelled by the comparison values defined by E_C when this comparison value is not neutral.*

We call a "closeness edge" an edge that is labelled by a comparison value in $V^C_{close} \cup \{always\}$, and "farness edge" an edge that is labelled by a comparison value in $V^C_{far} \cup \{never\}$,

Example 23 (Criterion graph)**.** *Let us consider the problem of partitioning the object set $\mathbb{O} = \{a, b, c, d, e\}$ according to a criterion C_1. We represent this problem by the criterion graph G_{C_1}, represented on Figure 5.1.*

The labels of G_{C_1} edges correspond to the comparison values between objects of $\mathbb{O}$ according to criterion C_1.

For example, the edge (a, c) labelled $+$ according to C_1 means that the comparison value between a and c according to C_1 is $+$ (denoted also $C_1(a, c) = +$ as we saw in Section 4.2.1).

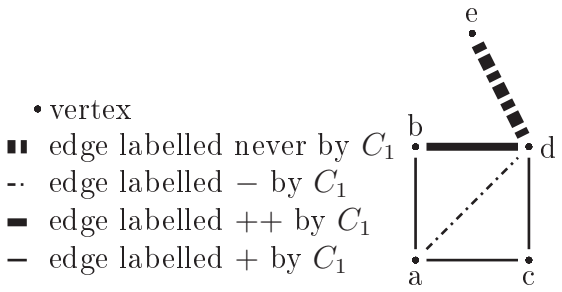

Figure 5.1: Criterion graph G_{C_1}

We notice that there is no edge between b and e, which means that $C_1(b, e) =$ *neutral.*

This definition can be generalised to the representation of partitioning an object set according to a criterion set. This problem is represented by a multi-graph with labelled edges.

Definition 9 (Criteria (set) graph)**.** *Let* $\mathbb{C}$*, be a criterion set on an object set* $\mathbb{O}$ *such as every criterion* C_i *in* $\mathbb{C}$ *is represented by a criterion graph* $G_{C_i} = (S, E_{C_i})$*. The multigraph* $\mathbb{G}_{\mathbb{C}} = (S, \bigcup E_{C_i} \forall C_i \in \mathbb{C})$ *is the criteria graph of* $\mathbb{C}$*, denoted* $\mathbb{G}_{\mathbb{C}} = (S, E_{\mathbb{C}})$*.*

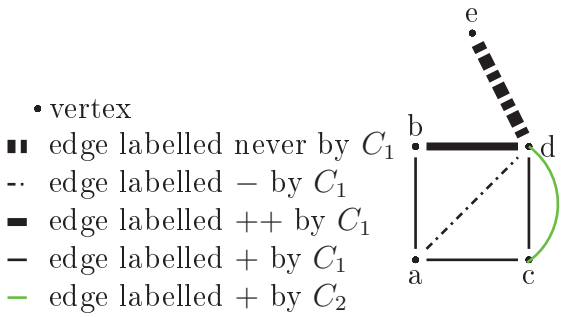

Figure 5.2: Criterion graph $G_{\{C_1,C_2\}}$

Example 24 (Criteria (set) graph)**.** *In Example 23, we represented the partitioning problem of the object set* $\mathbb{O} = \{a, b, c, d, e\}$ *according to the criterion* C_1 *by the criterion graph* G_{C_1} *on Figure 5.1.*

Let us represent the partitioning problem of the object set $\mathbb{O} = \{a, b, c, d, e\}$ *according to the criterion set* $\mathbb{C} = \{C_1, C_2\}$ *by the criterion graph* $G_{\{C_1,C_2\}}$ *on Figure 5.2.*

We can now represent our problem as a graph partitioning problem. Before defining the local and global partitioning semantics, we require to define some notions about partition values that are more specific to both global and local semantics.

5.1.3 About partition values

Local and global semantics are based on symbolic criteria (detailed in Section 4.2.1) and separate partitions into two groups: the **valid partitions** and the partitions that are not valid.

Intuitively, a valid partition is a partition such that there are not two objects with an *always* comparison value (respectively *never*) returned in distinct classes (respectively in a same class) of the partition. Let us define them formally according to one or several criteria. This valid partition notion is similar to [2].

Definition 10 (Valid partition according to a single criterion)**.** *Let P be a partition on an object set $\mathbb{O}$. P is valid according to the criterion C if and only if there are not two objects o_i, o_j in $\mathbb{O}$ such as:*

- $C(o_i, o_j) = never$ *and* $class(c_i, P) = class(c_j, P)$, *or*
- $C(o_i, o_j) = always$ *and* $class(c_i, P) \neq class(c_j, P)$.

Definition 11 (Valid partition)**.** *Let P be a partition on an object set $\mathbb{O}$. P is valid according to the criterion set $\mathbb{C}$ if and only if P is valid according to each criterion $C \in \mathbb{C}$ (please see Definition 10).*

Example 25 (Valid partition)**.** *In Example 24, we presented the criteria graph $G_{\{C_1,C_2\}}$ on Figure 5.2.*

The partition $\{\{a, b, c, d\}, \{e\}\}$ is a valid partition on $\mathbb{O}$ according to the criterion set $\mathbb{C} = \{C_1, C_2\}$.

The partition $P = \{\{a, b, c\}, \{d, e\}\}$ is not valid according to the criterion set $\mathbb{C} = \{C_1, C_2\}$ because $C_1(d, e) = never$ and $class(d, P) = class(e, P)$.

In the reminder of this we solely consider valid partitions.

Both global and local semantics first check if the partition is valid. If it is the case, they are interested in **incoherences**. Of course, the ideal case consists in never having objects in the same class with a farness value, respectively objects in different classes with a closeness value. If such a partition does not exist, there are *incoherences* with respect to the criteria set.

The incoherence (property of an object subset such that they must be in a same class according to some criteria and must be in separated classes according to other criteria) notion is central to the definitions of the two semantics detailed in Sections 5.2 and 5.3.

Example 26 (Incoherence in a criteria graph)**.** . *The criteria graph represented on Figure 5.1 present some incoherences. For example, a and d must be in distinct classes of a partition because* $C_1(a,d) = -$ *but they also must be in the same class because both must be into the same class as c. In fact, we have* $C_1(a,c) = +$*, and* $C_2(c,d) = +$*.*

In case of incoherences, from the point of view of a single criterion at a time, we prefer to group the objects linked by a stronger closeness comparison value than a weaker one. Similarly, we prefer to separate objects with a stronger farness value than objects with a weaker farness value. In the case of incoherences one has to make choices in order to satisfy closeness or farness values that, by definition, could lead to distinct partitions. Details will be exposed in Section 5.2.1. In other words, we would like to minimize the maximal intensity of **unsatisfied edges** labels. As it is defined in Definition 12, unsatisfied edges for a given partition are the farness labelled edges between two vertexes of the same class or the closeness labelled edges between two vertexes of distinct classes. This notion has been defined by [4].

Definition 12 (Satisfied edge)**.** *Let* G_C *be a criterion graph on an object set* $\mathbb{O}$*,* P *a partition on* $\mathbb{O}$ *and* o_i*,* o_j *two objects in* $\mathbb{O}$*.* (o_i, o_j) *is a satisfied edge for* G_C *according to the partition* P *on* $\mathbb{O}$ *if and only if:*

- $class(o_i, P) = class(o_j, P)$ *and* $c(o_i, o_j) \in V^C_{close} \cup \{always\}$*, (called closeness satisfied edges) or*
- $class(o_i, P) \neq class(o_j, P)$ *and* $c(o_i, o_j) \in V^C_{far} \cup \{never\}$ *(called farness satisfied edges).*

If (o_i, o_j) *is not a satisfied edge,* (o_i, o_j) *is said a (closeness or farness) unsatisfied edge.*

If an edge is unsatisfied for a partition P *on a criterion graph* G_C*, it is also unsatisfied for* P *on a criteria graph* $\mathbb{G}_\mathbb{C}$ *such as* $C \in \mathbb{C}$*.*

Example 27 (Unsatisfied edge)**.** *Let us consider the partition* $P_4 = \{\{a, c, d\}, \{b, e\}\}$ *of Example 28 on the graph* G_C *represented on Figure 5.1.*

Let us identify the unsatisfied edges on G_C *according to* P_4*:*

- *unsatisfied closeness edges:* (b,d)*,* (a,b) *(because b is not in the same class as a and d and* (b,d)*,* (a,b) *are closeness edges).*
- *unsatisfied farness edge:* (a,d) *(because a is in the same class than d and* (a,d) *is a closeness edge).*

The local semantics improves global semantics by the way it takes care of incoherences. This will be detailed in Section 5.3. Let us start by global semantics in the following Section.

5.2 Global semantics

In this section, we will see how to evaluate and compare partitions according to global semantics (Sections 5.2.1 and 5.2.2) with respect to one or several criteria. Once it is done, we will present algorithms to find best partition values according to global semantics and with respect to one criterion (Section 5.2.3) or several criteria (Section 5.2.4).

Let us evaluate and compare partitions on a same set of objects $\mathbb{O}$ according to global semantics in order to determine what is a best partition according to global semantics.

We remind ourselves that from the point of view of any criterion, we prefer to group objects linked by a stronger closeness comparison value than a weaker one. Similarly, we prefer to separate objects with a stronger farness value than objects with a weaker farness value.

We will start by doing it with respect to a single criterion before doing it with respect to several criteria.

5.2.1 Evaluating and comparing partitions according to global partitioning semantics for a single criterion

Let us remind ourselves that we are interested only into partitions that never put strictly different objects in the same class and always put identical objects in the same class. Those partitions are called valid as seen in Definition 11. Not valid partitions are simply assigned by the semantics a $notValid$ partition value.

Once a partition is valid, the value of a partition is composed of two values: the **inter (class) value** (denoted v_p, with p for "positive") and the **intra (class) value** (denoted v_n, with n for "negative"). The inter value v_p is the weakest (please see Definition 13) closeness or *always* value such that all object pairs with a stronger or equivalent comparison value to v_p are in the same class (denoted "satisfied" edge in [4], please see Definition 12). The bipolar condition on the intra value v_n also applies: v_n is the weakest farness or *never* value such that all object pairs with a stronger or equivalent comparison value to v_n are in distinct classes.

In other words, the inter class value concerns proximity between classes ("split of classes" criterion [33][32]), and the intra class value concerns the farness into classes (homogeneity criterion [33][32]).

The partition value of P according to C is denoted $v(P, C) = (v_p, v_n)$. In order to formally define it in Definition 14, we require to define the weakest value of a set of comparable comparison values in Definition 13.

Definition 13 (Weakest value)**.** *Let $\mathbb{V}$ be a comparison value set such that all pairs of values v_1, v_2 in $\mathbb{V}$ are comparable together. The value v such that $v \in \mathbb{V}$ and $v \leq v_i \forall v_i \in \mathbb{V}$ is said the weakest value of $\mathbb{V}$. The weakest value v of $\mathbb{V}$ is denoted $weakest(\mathbb{V})$.*

Definition 14 (Partition value with respect to a single criterion)**.** *Let $G_C = (S, E)$, be the criterion graph of criterion C, and P a partition on G_C. In the case where P is not valid (Definition 10), then P's value is notValid. If P is valid, P's value is (v_p, v_n) with :*

- $v_p = weakest((V^C_{close} \cup \{always\}) \cap \{v | \forall$ *edge* (s_i, s_j) *labelled* v, $classe(s_i) = classe(s_j)\})$
- $v_n = weakest((V^C_{far} \cup \{never\}) \cap \{v | \forall$ *edge* (s_i, s_j) *labelled* v, $classe(s_i) \neq classe(s_j)\})$

We denote $v(P, C)$ the P partition value with respect to criterion C (abusively $v(P)$ if C is obvious).

Example 28 (Partition value according to a single criterion)**.** *Let us consider the criterion graph G_{C_1} of Example 23 that is represented on Figure 5.1. For this criterion, there are two closeness values ($V^C_{close} = \{+, ++\}$) and a single farness value ($V^C_{far} = \{-\}$). Let us evaluate several partitions on $\mathbb{O} = \{a, b, c, d, e\}$ according to C_1:*

- $P_1 = \{\{a, c\}, \{b, d, e\}\}$. $v(P_1, C_1) = notValid$. *This partition is not valid because d and e are in a same class and $C_1(d, e) = never$.*
- $P_2 = \{\{b, d, c\}, \{a\}, \{e\}\}$. $v(P_2, C_1) = (++, -)$. *The inter value is only $++$ because a and c are in distinct classes but $C_1(a, c) = +$.*
- $P_3 = \{\{b, d\}, \{a, c\}, \{e\}\}$. $v(P_3, C_1) = (++, -)$. *The inter value is only $++$ because d and e are in distinct classes but $C_1(d, c) = +$.*
- $P_4 = \{\{a, d, c\}, \{b, e\}\}$. $v(P_4, C_1) = (always, never)$. *$b$ and d are in distinct classes despite $C_1(b, d) = ++$, so the intra value can only be always. However, this partition is valid because d and e are also in distinct classes.*
- $P_5 = \{\{a, b, c, d\}, \{e\}\}$. $v(P_5, C_1) = (+, never)$. *The inter value is never because a and d are in the same class but $C_1(a, d) = -$.*

Let us see how partition values are compared in order to compare partitions to each other. A partition P_i is better than another one P_i according to a criterion C if they share the same object set and both intra and inter values of P_i are weaker than intra and inter values of P_j according to C.

A weakest intra value intuitively means that more levels of closeness values of C are satisfied, and each stronger levels of closeness values are satisfied. If P_j and P_i have the same intra and inter values, their values are equal. If their values are not equal but P_j has a weaker intra value than P_i and P_i has a weaker inter value than P_j, their values are incomparable and so are P_i and P_j.

Definition 15 (Partition order for a single criterion). *Let P_i, P_j be two valid partitions on a criterion graph G_C such as their partition values are respectively (v_p, v_n) and (v'_p, v'_n).*

- *(v_p, v_n) is equal to (v'_p, v'_n) (denoted $(v_p, v_n) = (v'_p, v'_n)$) if and only if $v_p = v'_p$ and $v_n = v'_n$.*
- *(v_p, v_n) is better than (v'_p, v'_n) (denoted $(v_p, v_n) \succeq (v'_p, v'_n)$) if and only if $v_p \leq v'_p$ and $v_n \geq v'_n$.*
- *(v_p, v_n) is strictly better than (v'_p, v'_n) (denoted $(v_p, v_n) \succ (v'_p, v'_n)$) if and only if $(v_p, v_n) \succ (v'_p, v'_n)$ and $(v_p, v_n) \neq (v'_p, v'_n)$.*
- *in other cases, (v_p, v_n) and (v'_p, v'_n) are not comparable (denoted $(v_p, v_n) \diamond (v'_p, v'_n)$).*

The partition order follows their partition values order.

Example 29 (Partition values order for a single criterion). *Let us consider the criterion graph G_{C_1} of Example 23 that is represented on Figure 5.1. Let us consider the partitions evaluated in Example 28 according to the criterion C_1:*

- $v(P_1, C_1) = notValid$.
- $v(P_2, C_1) = (++, -)$.
- $v(P_3, C_1) = (++, -)$.
- $v(P_4, C_1) = (always, never)$.
- $v(P_5, C_1) = (+, never)$.

P_1 is not valid, so comparing P_1 with another partition has no meaning (P_1 is worse).

P_2 and P_3 have the same value: P_2 is equivalent to P_3 (denoted $P_2 \approx P_3$).

P_3 has a weaker value than P_5 ($- < never$) but a stronger inter value than P_5 ($++ > +$), so P_3 and P_5 are not comparable ($P_3 \diamond P_5$).

P_5 has a better value than P_4 because both intra and inter values of P_3 are weaker than intra and inter values of P_4 ($+ < always$ and $never \leq never$, so $v(P_5, C_1) \preceq v(P_4, C_1)$), so P_5 is preferred to P_4 (denoted $P_5 \preceq P_4$).

In this Section, we saw how to evaluate and compare partitions according to global semantics and a single criterion. Let us extend it to several criteria in the following Section.

5.2.2 Evaluating and comparing partitions according to global partitioning semantics for several criteria

Let us present the evaluation of a partition according to global semantics and with respect to several criteria in order to give an order on partitions according to global semantics and with respect to several criteria.

Definition 16 (Partition value with respect to a criterion set)**.** *Let $\mathbb{G}_{\mathbb{C}} = (S, E)$, be the criteria graph of the criterion set $\mathbb{C}$, and P a partition on $\mathbb{G}_{\mathbb{C}}$. In the case where P is not valid (Definition 11), then P's value is notValid.*

If P is valid, P's value according to $\mathbb{C}$ is the set of P's values according to each criterion $C \in \mathbb{C}$.

We denote $v(P, \mathbb{C}) = v(P, \mathbb{C}) = \{C_{C(P,C)} | C \in \mathbb{C}\}$ the P partition value with respect to criterion set $\mathbb{C}$ (abusively $v(P)$, if $\mathbb{C}$ is obvious).

Example 30 (Partition value according to a criterion set)**.** *Let us consider the criterion graph $G_{\{C_1,C_2\}}$ of Example 24 that is represented on Figure 5.2. We evaluated some partitions in Example 28 with respect to criterion C_1. Let us evaluate them according to $\mathbb{C} = \{C_1, C_2\}$ with C_2 such as $V^{C_2}_{close} = \{+\}$ and $V^{C_2}_{far} = \{\}$:*

- $P_1 = \{\{a, c\}, \{b, d, e\}\}$. $v(P_1, \mathbb{C}) = notValid$.
- $P_2 = \{\{b, d, c\}, \{a\}, \{e\}\}$. $v(P_2, \mathbb{C}) = \{C_{1(++,-)}, C_{2(+,never)}\}$.
- $P_3 = \{\{b, d\}, \{a, c\}, \{e\}\}$. $v(P_3, \mathbb{C}) = \{C_{1(++,-)}, C_{2(always,never)}\}$.
- $P_4 = \{\{a, c, d\}, \{b, e\}\}$. $v(P_4, \mathbb{C}) = \{C_{1(always,never)}, C_{2(+,never)}\}$.
- $P_5 = \{\{a, b, c, d\}, \{e\}\}$. $v(P_5, \mathbb{C}) = \{C_{1(+,never)}, C_{2(+,never)}\}$.

If there are several criteria, we consider according to Work Hypothesis 6 page 19 that criteria are independent [14] from each other, which means that no criterion is strictly more important than another to decide which partition is best.

Let A, B, be two independent criteria (denoted $A \diamond B$) defined on the same object set $\mathbb{O}$. To compare two partitions with respect to $\{A, B\}$, the comparison of their values with respect to A is as important as the comparison of their values with respect to B. Let $\mathbb{C}$ be a criterion set so that criteria are all independent from each other (namely an independent criterion set). Then for all distinct $C_i, C_j \in \mathbb{C}$ we have $C_i \diamond C_j$.

A partition P has a best partition value according to a criterion set $\mathbb{C}$ if P is valid and it is impossible to improve an inter or intra value of one $\mathbb{C}$ criterion without decreasing an inter or intra value of at least a $\mathbb{C}$ criterion. A best partition value corresponds to a Pareto equilibrium [79] among criteria values. Indeed, a Pareto equilibrium is obtained when it is impossible to improve the satisfaction of something (a criterion here) without decreasing the satisfaction of something else.

Let us formally see how to compare two partitions with respect to an independent criterion set in Definition 17.

Definition 17 (Partition values order with respect to an independent criterion set)**.** *Let P_1, P_2 be two partitions on $\mathbb{G}_{\mathbb{C}}$, a criteria graph such that $\mathbb{C}$ is an independent criterion set.*

The P_1 partition value is better than P_2 partition value (denoted $v(P_1, \mathbb{C}) \succeq v(P_2, \mathbb{C})$) if and only if $\forall C_i \in \mathbb{C}, v(P_1, C_i) \succeq v(P_2, C_i)$.

The P_1 partition value is strictly better than P_2 partition value (denoted $v(P_1, \mathbb{C}) \succ v(P_2, \mathbb{C})$) if and only if:

- $\forall C_i \in \mathbb{C}$, $v(P_1, C_i) \succeq v(P_2, C_i)$ *and*
- $\exists C_j \in \mathbb{C} | v(P_1, C_j) \succ v(P_2, C_j)$.

Values of partitions P_1 and P_2 are incomparable for $\mathbb{G}_{\mathbb{C}}$ in other cases.

Example 31 (Partition values order with respect to an independent criterion set)**.** *Let us consider the criteria graph $G_{\{C_1,C_2\}}$ of Example 24 that is represented on Figure 5.2. Let us consider the partitions evaluated in Example 30 according to the criterion set $\mathbb{C} = \{C_1, C_2\}$.*

- $P_1 = \{\{a, c\}, \{b, d, e\}\}$. $v(P_1, \mathbb{C}) = notValid$.
- $P_2 = \{\{b, d, c\}, \{a\}, \{e\}\}$. $v(P_2, \mathbb{C}) = \{C_{1(++,-)}, C_{2(+,never)}\}$.
- $P_3 = \{\{b, d\}, \{a, c\}, \{b\}, \{e\}\}$. $v(P_3, \mathbb{C}) = \{C_{1(++,-)}, C_{2(always,never)}\}$.
- $P_4 = \{\{a, e\}, \{b\}, \{c, d\}\}$. $v(P_4, \mathbb{C}) = \{C_{1(always,never)}, C_{2(+,never)}\}$.
- $P_5 = \{\{a, b, c, d\}, \{e\}\}$. $v(P_5, \mathbb{C}) = \{C_{1(+,never)}, C_{2(+,never)}\}$.

P_1 is not valid, so comparing P_1 with another partition has no meaning.

P_2 and P_3 have the same value according to C_1 but P_2 has a better value than P_3 according to C_2 so $P_2 \succeq P_3$ according to $\mathbb{C}$.

P_3 is not comparable to P_5 with respect to C_1, so P_3 and P_5 are also not comparable ($P_3 \diamond P_5$) with respect to $\mathbb{C}$.

P_5 has a better value than P_4 because it has a better value according to both C_1 and C_2 criteria.

In this section, we saw how to evaluate and compare partitions according to global semantics and with respect to one or several criteria. Let us see which algorithms are used to find best partitions according to global semantics.

5.2.3 Finding best partitions for a single criterion

Let C be a criterion on an object set $\mathbb{O}$. In order to find the best partition values on $\mathbb{O}$ with respect to C, we have to find and to evaluate reference partitions on $\mathbb{O}$ with respect to C for all closeness values $v_i \in V^C_{close} \cup \{always\}$. To define a **reference partition** we need to use the notion of a **refined partition** as explained below.

Definition 18 (Refined partition). *Let P_i, P_j be two partitions on an object set $\mathbb{O}$. P_i is more refined than P_j if and only if $\forall\ c_i$ class $\in P_i\ \exists c_j$ class $\in P_j | c_i \subseteq c_j$. The partition P_j is said to be less refined than P_i partition.*

Example 32 (Refined partition). *Let $\mathbb{O}$ be an object set such as $\mathbb{O} = \{a, b, c, d, e\}$.*

The partition $P_a = \{\{a, b, c\}, \{d, e\}\}$ is more refined than the partition $P_a = \{\{a, b, c, d, e\}\}$ but both are less refined than $P_c = \{\{a, b\}, \{c\}, \{d, e\}\}$.

However, P_a is not more refined than $P_d = \{\{a, b\}, \{c, d\}, \{e\}\}$ because no class of P_a includes $\{c, d\}$ and P_d is also no more refined than P_a because no class of P_d includes $\{d, e\}$.

Definition 19 (Reference partition for a criterion). *Let C be a criterion on an object set $\mathbb{O}$ and v_i a comparison value such that $v_i \in V^C_{close} \cup \{always\}$. The reference partition P_{ref} for C with respect to v_i is the most refined partition P such as $v(P, C) = (v_p, v_n)$ and $v_p \leq v_i$.*

Let G_C be the criterion graph on $\mathbb{O}$ with respect to C criterion. We denote $ref(v_i, G_C)$ the reference partition for a criterion C with respect to the closeness value v_i (or abusively $ref(v_i)$ when G_C is obvious).

Example 33 (Reference partition for a criterion). *Let us consider the criterion graph G_{C_1} in Example 23 that is represented on Figure 5.1. Let us consider the closeness value set of the criterion C_1: $V^{C_1}_{close} = \{+, ++\}$. Let us present reference partitions on $\mathbb{O} = \{a, b, c, d, e\}$ according to C.*

- *$P_+ = ref(+) = \{\{a, b, c, d\}, \{e\}\}$. a,b,c and d are in the same class because of the comparisons values $C(d, b) = ++$ and $C(a, b) = C(a, c) = C(c, d) = +$.*
- *$P_{++} = ref(++) = \{\{a\}, \{b, d\}, \{c\}, \{e\}\}$. b and d are in the same class because of the comparison value $C(d, b) = ++$.*
- *$P_a = ref(always) = \{\{a\}, \{b\}, \{d\}, \{c\}, \{e\}\}$. No objects are in the same class because there are not two vertexes such that their comparison value according to C is always or a stronger comparison value (no comparison stronger than always exists).*

Those reference partitions are represented on Figure 5.3. We remark that the stronger the taken "as reference" value is, the more the partition is refined. This is due to the fact that less closeness comparison values require to put some object pairs in a same class.

Evaluating the reference partitions is enough to calculate and evaluate all the best partition values. Since the reference partition with respect to a closeness or an *always* value v_i for criterion C is the most refined partition with $v_p \leq v_i$, it is the partition with the less possible unsatisfied farness edges (farness edges such as both vertexes are inside the same class, as explained in Definition 12). This makes it a partition with the best possible v_n value with respect to the v_p such that it is $\leq v_i$.

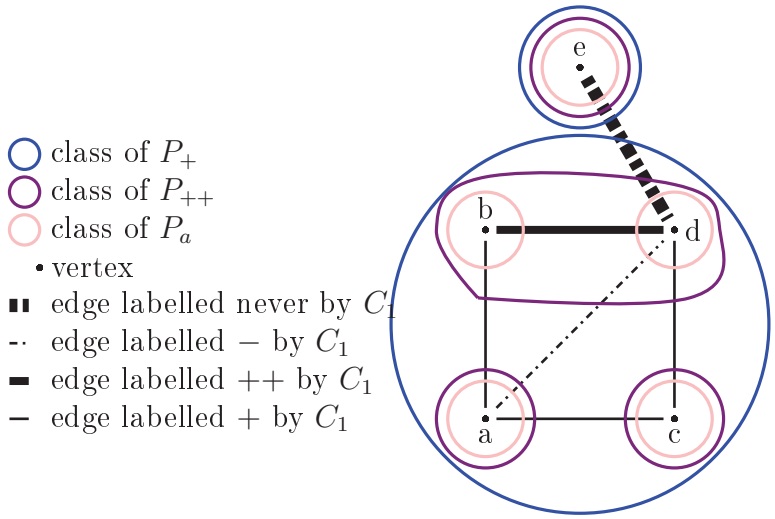

Figure 5.3: Reference partitions for G_{C_1}

Because a partition value has a v_p in $V^C_{close} \cup \{always\}$, calculating the values of reference partitions for each v_i in $V^C_{close} \cup \{always\}$ is enough to determine all best partition values.

To calculate a reference partition with respect to a closeness value v_i for a criterion C comes to calculating connected components on G_C considering only v'_i labelled edges such as $v'_i \geq v_i$. We can then simply use a Kruskal's algorithm[50]. This algorithm is used to find the shortest spanning tree of a graph based on connected components. According to [23], the complexity of Kruskal's algorithm is $\mathcal{O}(m \log n)$ for n vertexes and m edges if the connected components are implemented as a disjoint-set data structure. Algorithm 2 presents an adaptation of Kruskal's algorithm in order to find the connected components of a graph instead of the shortest spanning tree. In our case, the set of connected components is implemented in a table such as the indices correspond to the vertexes to partition and the values to the classes affected to the objects. So, merging two connected components has a complexity of $\mathcal{O}(n)$ for n vertexes (each vertex must be checked in order to determine if its connected component has changed) and find the class of a vertex has a complexity of $\mathcal{O}(1)$. Since there are n connected components at the beginning, there are $n-1$ merges of connected components at most because each merge decreases of 1 the number of connected components. So, the merge (complexity $\mathcal{O}(n)$ in our implementation) line 6 is done $n-1$ times at most. Each edge (m) is also checked line 4. As a consequence, the complexity of calculating a reference partition is, in our implementation, $\mathcal{O}(m + n^2)$. However, the complexity is $\mathcal{O}(m \log n)$ with an appropriate implementation, as mentioned above. So, we will consider $\mathcal{O}(m \log n)$ as the complexity of calculating a reference partition in the remainder of this thesis.

Please note that the connected component idea has been already explored in [32]

and [4]. However the authors do not consider incoherence problems or even more levels of farness and closeness values.

If a tested reference partition P_{ref} has a value (v_n, v_p) such as v_p is the weakest value in $V^C_{far} \cup \{never\}$, then the reference partitions according to a value $v'_p > v_p$ will have a partition value worse than P_{ref} and there is no need to evaluate and test them. Algorithm 1 ends in this case because of the test line 8.

Algorithm 1 BestValuesForASingleCriterion

Require: C: criterion on an object set $\mathbb{O}$; G_C: criterion graph of C on $\mathbb{O}$;
Ensure: set of best partition values with respect to C on $\mathbb{O}$

```
1:  best partition value set bestV = {};
2:  for all values v_i ∈ V^C_close ∪ {always} in < order do
3:    Partition P= ref(v_i);
4:    Partition value v = v(P);
5:    if P is valid and ∄v' ∈ bestV | v' ⪰ v then
6:      add v to bestV;
7:    end if
8:    if v(P) = (v'_p, v'_n) such as v_n = weakest(V^C_far ∪ {never}) then
9:      return bestV;
10:   end if
11: end for
12: return bestV;
```

Let us see an Example about the results of Algorithm 1.

Example 34 (Results of Algorithm 1)**.** *Let us consider the criterion graph G_{C_1} on the object set $\mathbb{O} = \{a, b, c, d, e\}$ defined Example 23. In Example 33 we gave all its reference partitions, represented on Figure 5.3. Let us apply Algorithm 1 on G_{C_1}. We remind ourselves that $V^C_{close} = \{+, ++\}$ and $V^C_{far} = \{-\}$.*

- *$+$ is the weakest comparison value in $V^C_{close} \cup \{always\}$, so the first reference partition calculated and evaluated is $P_+ = ref(+) = \{\{a, b, c, d\}, \{e\}\}$. $v(P_+) = (+, never)$ and is a best partition value because $v(P_+)$ is valid and there are not best partition values to be compared with yet.*

- *$++$ is the weakest comparison value in $V^C_{close} \cup \{always\} - \{+\}$, so the first calculated and evaluated reference partition is $P_{++} = ref(++) = \{\{a\}, \{b, d\}, \{c\}, \{e\}\}$. $v(P_{++}) = (++, -)$. This value is not comparable to $v(P_+)$, so it is also a best value. $-$ is the weakest value in $V^C_{far} \cup \{never\}$, so the algorithm ends (without calculating and evaluating $ref(always)$) and returns the set of founded best values: $\{(+, never), (++, -)\}$.*

In Example 28, we evaluated five partitions on G_{C_1}. Partitions P_2 and P_3 are best partitions because they both have a best value $(++, -)$. Partition P_5 is also a best partition because it has a best value $(+, never)$.

Complexity. In the worst case (when there is no valid partition with a v_n value such that $v_n = max(V^C_{far} \cup \{never\})$) we have $k+1 = |V^C_{close} \cup \{always\}|$ reference partitions to find and evaluate. The complexity of the global semantics for one criterion algorithm is $\mathcal{O}((k+1) * m \log n)$, and it is depicted below (Algorithm 1).

In this Section, we saw how to calculate every best partition value on an object set $\mathbb{O}$ and according to a criterion C. Let us extend this to several criteria.

5.2.4 Finding best partitions for several criteria

Let us now consider the global semantics when there are more than one criterion to consider. We will first need three notions: **closeness value set**, **ascendant closeness value set** and **reference partition**. The reference partitions, as explained in Section 5.2.3, are the actual tests to be performed by the algorithm. The cardinality of the closeness value set represents the number of tests that the algorithm will need to perform in the worst case. Finally, the ascendant closeness values notion will allow us to skip some tests, and optimise the algorithm.

Definition 20 (Closeness value set)**.** *A closeness value set $\mathbb{VC}$ for a criterion set $\mathbb{C}$ is a set of comparison values v_i such as $v_i \in V^{C_i}_{close} \cup \{always\}$ and $C_i \in \mathbb{C}$, with one and only one comparison value v_i for each criterion $C_i \in \mathbb{C}$.*

Example 35 (Closeness value set)**.** *In Example 24, we defined the criteria graph $\mathbb{G}_{\{C_1,C_2\}}$ with C_1, C_2 such as $V^{C_1}_{close} = \{++, +\}$ and $V^{C_2}_{close} = \{+\}$. Let us enumerate the closeness value set according to $\{C_1, C_2\}$:*

- $\mathbb{VC}_1 = \{C_{1,+}, C_{2,+}\}$
- $\mathbb{VC}_2 = \{C_{1,+}, C_{2,always}\}$
- $\mathbb{VC}_3 = \{C_{1,++}, C_{2,+}\}$
- $\mathbb{VC}_4 = \{C_{1,++}, C_{2,always}\}$
- $\mathbb{VC}_5 = \{C_{1,always}, C_{2,+}\}$
- $\mathbb{VC}_6 = \{C_{1,always}, C_{2,always}\}$

Definition 21 (Ascendant closeness value set)**.** *Let $\mathbb{VC}_1$ and $\mathbb{VC}_2$ be two closeness value sets for the same criterion set $\mathbb{C}$. $\mathbb{VC}_1$ is an ascendant of $\mathbb{VC}_2$ if and only if $\mathbb{VC}_1$ has a weaker comparison value than $\mathbb{VC}_2$ for each criterion in $\mathbb{C}$.*

$\mathbb{VC}_2$ is said a descendant of $\mathbb{VC}_1$.

Example 36 (Ascendant closeness value set)**.** *In Example 35, we enumerated all closeness value sets for C_1, C_2 such as $V^{C_1}_{close} = \{++, +\}$ and $V^{C_2}_{close} = \{+\}$. Let us determine their ascendant relations.*

- $\mathbb{VC}_1 = \{C_{1,+}, C_{2,+}\}$ *has the weakest possible value for both* C_1 *and* C_2*, so* $\mathbb{VC}_1$ *is an ascendant of all the other closeness value sets (*$\mathbb{VC}_2$,$\mathbb{VC}_3$,$\mathbb{VC}_4$,$\mathbb{VC}_5$ *and* $\mathbb{VC}_6$*).*

- $\mathbb{VC}_2 = \{C_{1,+}, C_{2,always}\}$ *is an ascendant of* $\mathbb{VC}_4$ $\mathbb{VC}_6$*. In fact,* $\mathbb{VC}_3$ *and* $\mathbb{VC}_5$ *have a weaker value (+) than* $\mathbb{VC}_2$ *for* C_2 *(always).*

- $\mathbb{VC}_4 = \{C_{1,++}, C_{2,always}\}$ *is a descendant of* $\mathbb{VC}_2$ *and an ascendant of* $\mathbb{VC}_6$*.*

- $\mathbb{VC}_6 = \{C_{1,always}, C_{2,always}\}$ *is a descendant of all other closeness value sets.*

Definition 22 (Reference partition with respect to a criterion set)**.** *Let* $\mathbb{C}$ *be a criterion set on an object set* $\mathbb{O}$*, and* $\mathbb{VC}$ *a closeness value set for* $\mathbb{C}$*. The reference partition* P_{ref} *for* $\mathbb{C}$ *with respect to* $\mathbb{VC}$ *(denoted* $ref(\mathbb{VC})$*) is the most refined (please see Definition 18) partition such as* $v(P_{ref}) = \{v(P_{ref}, C_i) \forall C_i \in \mathbb{C}\}$ *with* $\forall C_i$ *criterion:* $v(P_{ref}, C_i) = (v_p, v_n) | v_p \leq v_i \in \mathbb{VC}$*.*

Example 37 (Reference partition with respect to a criterion set)**.** *In Example 33, we calculated reference partitions for the criterion graph* G_{C_1} *defined Example 23 and represented on Figure 5.3. Let us enumerate the reference partitions for the criteria graph* $\mathbb{G}_{\{C_1,C_2\}}$ *with respect to the closenes value set enumerated in Example 35:*

- $P_{+,+} = ref(\{C_{1,+}, C_{2,+}\}) = \{\{a,b,c,d\},\{e\}\}$

- $P_{+,a} = ref(\{C_{1,+}, C_{2,always}\}) = \{\{a,b,c,d\},\{e\}\} = P_{+,+}$*.* $P_{+,+} = P_{+,a}$ *is exactly the reference partition* P_+ *on the graph* G_C *(Example 33) and is represented on Figure 5.3.*

- $P_{++,+} = ref(\{C_{1,++}, C_{2,+}\}) = \{\{a\},\{b,d,c\},\{e\}\}$*.*

- $P_{++,a} = ref(\{C_{1,++}, C_{2,always}\}) = \{\{a\},\{b,d\},\{c\},\{e\}\}$*.* $P_{++,a}$ *is exactly the reference partition* P_{++} *on the graph* G_C *(Example 33) and is represented on Figure 5.3.*

- $P_{a,+} = ref(\{C_{1,always}, C_{2,+}\}) = \{\{a\},\{b\},\{d,c\},\{e\}\}$*.*

- $P_{a,a} = ref(\{C_{1,always}, C_{2,always}\}) = \{\{a\},\{b\},\{d\},\{c\},\{e\}\}$*.* $P_{a,a}$ *is exactly the reference partition* P_a *on the graph* G_C *(Example 33) and is represented on Figure 5.3.*

The global semantics algorithm for several criteria is the extension for the algorithm for one criterion (Algorithm 1). The best partition values are also reference partition values, so we calculate, evaluate and compare them.

First, we find all closeness value set (Definition 20) for $\mathbb{C}$. We compute the reference partition (Definition 22) for each closeness value set $\mathbb{VC}$ by searching for

connected components with Kruskal's algorithm (complexity $\mathcal{O}(m \log n)$ [2]) on $\mathbb{G}_{\mathbb{C}}$ considering only v_p labelled edges such that $v_p \geq v_i | v_i \in (V_{close}^{C_i} \cup \{always\}) \cap \mathbb{VC}$ and $C_i \in \mathbb{C}$. We then evaluate reference partition values and only keep best ones.

If a reference partition $ref(\mathbb{VC})$ has $v(P, C_i) = (v_p, v_n)$ for each criterion $C_i \in \mathbb{C}$ such as $v_n = weakest(V_{far}^{C_i} \cup \{never\})$, then the reference partitions $ref(\mathbb{VC}')$ with $\mathbb{VC}'$ descendants (Definition 21) of $\mathbb{VC}$ have a worse or same value than $ref(\mathbb{VC})$, so we do not need to evaluate them.

Algorithm 2 ConnectedComponent

Require: $G(V, E)$: graph with V, the vertex set and E, the edge set;
Ensure: the partition on V such that its classes corresponds to connected components of $G(V, E)$

1: **for all** vertex $v \in V$ **do**
2: $connectedComponentOf(v) = \{v\}$;
3: **end for**
4: **for all** edge $(x, y) \in E$ **do**
5: **if** $connectedComponentOf(x) \neq setOf(y)$ **then**
6: merge $connectedComponentOf(x)$ and $connectedComponentOf(y)$;
7: **end if**
8: **end for**
9: **return** the set of the $connectedComponentOf(v)$;

Algorithm 3 globalAlgorithm

Require: $\mathbb{C}$, a criterion set on an object set $\mathbb{O}$; $\mathbb{G}_{\mathbb{C}}$ criteria graph of $\mathbb{C}$
Ensure: set of best partition values with respect to $\mathbb{C}$ on $\mathbb{O}$.

1: best partition value set $bestV = \{\}$;
2: set of closeness value set to test $toTest = \{\mathbb{VP} | \mathbb{VP}$, closeness value set for $\mathbb{C}\}$;
3: **while** $toTest \neq \{\}$ **do**
4: pick up $\mathbb{VP}$ from $toTest$ such as $\mathbb{VP}$ has no ascendant in $toTest$;
5: Partition $P = ref(\mathbb{VP})$;
6: Partition value $v = v(P, \mathbb{C})$;
7: **if** P is valid and $\nexists v' \in bestV | v' \succeq v$ **then**
8: add v to $bestV$;
9: **end if**
10: **if** $\forall C_i \in \mathbb{C}, v(P, C_i) = (v_p, v_n) | v_n = weakest(V_{far}^{C_i} \cup \{never\})$ **then**
11: remove all descendants of $\mathbb{VP}$ from $toTest$;
12: **end if**
13: **end while**
14: **return** $bestV$;

[2] with n vertexes and m edges

Example 38 (Results of Algorithm 3)**.** *In Example 34, we presented the results of Algorithm 1 applied on the criterion graph G_C. Let us apply Algorithm 3 on the criteria graph $\mathbb{G}_{\{C_1,C_2\}}$ (defined in Example 24 and represented on Figure 5.2) in order to find all best partition values on $\mathbb{G}_{\{C_1,C_2\}}$. First, we need the set of closeness value sets for $\{C_1, C_2\}$. All those closeness value sets are enumerated in Example 37: toTest = $\{\{C_{1,+}, C_{2,+}\}$, $\{C_{1,+}, C_{2,always}\}$, $\{C_{1,++}, C_{2,+}\}$, $\{C_{1,++}, C_{2,always}\}$, $\{C_{1,always}, C_{2,+}\}$, $\{C_{1,always}, C_{2,always}\}\}$.*

- *The only closeness value set of toTest that has no ascendant in toTest is $\{C_{1,+}, C_{2,+}\}$.*

  $ref(\{C_{1,+}, C_{2,+}\}) = \{\{a, b, c, d\}, \{e\}\}$, which has $\{C_{1(+,never)}, C_{2(+,never)}\}$ as partition value. There is no best value yet, so this one is a best value.

  $\{C_{1,+}, C_{2,+}\}$ is removed from toTest.

- *The closeness value set $\{C_{1,+}, C_{2,always}\}$ of toTest that has no ascendant in toTest.*

  $ref(\{C_{1,+}, C_{2,always}\}) = \{\{a, b, c, d\}, \{e\}\}$, which has $\{C_{1(+,never)}, C_{2(+,never)}\}$ as partition value. This partition value is already identified as a best one.

  $\{C_{1,+}, C_{2,always}\}$ is removed from toTest.

- *The closeness value set $\{C_{1,++}, C_{2,+}\}$ of toTest that has no ascendant in toTest.*

  $ref(\{C_{1,++}, C_{2,+}\}) = \{\{a\}, \{b, c, d\}, \{e\}\}$, which has $\{C_{1(++,-)}, C_{2(+,never)}\}$ as partition value. This partition value is not worse than the actual best partition value $\{C_{1(+,never)}, C_{2(+,never)}\}$, so it is also a best partition value.

  We notice that $weakest(V^{C_1}_{far} \cup \{never\}) = -$ and $weakest(V^{C_2}_{far} \cup \{never\}) = never$, so the test line 10 of Algorithm 3 result is "true" and we remove from totest $\{C_{1,++}, C_{2,+}\}$ and its descendants ($\{C_{1,++}, C_{2,always}\}$, $\{C_{1,always}, C_{2,+}\}$, $\{C_{1,always}, C_{2,always}\}\}$).

  toTest became empty and the algorithm returns the set of best partition values on $\mathbb{G}_{\{C_1,C_2\}}$: $\{\{C_{1(+,never)}, C_{2(+,never)}\}$, $\{C_{1(++,-)}, C_{2(+,never)}\}\}$.

Complexity. For a criterion set $\mathbb{C}$ of c criteria, we have to calculate and evaluate $|V^{C_1}_{close} \cup \{always\}| * ... * |V^{C_c}_{close} \cup \{always\}|$ reference partitions in the worst case, namely $(k+1)^c$ reference partitions with $k = max(|V^{C_i}_{close}| \ \forall C_i \in \mathbb{C})$. So, this algorithm has $\mathcal{O}((k+1)^c * m \log n)$ as complexity (please see Algorithm 3).

Let us show with the three following Examples that global semantics can give good qualitative results and that global semantics results are very dependent from the considered object set, which is fixed by local semantics.

Let us consider the Example in Table 5.2.4 that represents contextual entities and namely the contextual entities contained in the two Sudoc subsets related to

id	title	date	domain	appellation
Nc_1	"*Le banquet*"	"1868"		"PLATON"
Nc_2	"*Le banquet*"	"2007"		"PLATON"
Nc_3	"*Letter to a Christian nation*"		religion	"HARRIS, Sam"
Nc_4	"*Surat terbuka untuk bangsa kristen*"	"2008"	religion	"HARRIS, Sam"
Nc_5	"*The philosophical basis of theism*"	"1883"	religion	"HARRIS, Sam"
Nc_6	"*Building pathology*"	"2001"	building	"HARRIS, Samuel"
Nc_7	"*Building pathology*"	"1936"	building	"HARRIS, Samuel"
Nc_8	"*Aluminium alloys 2002*"	"2002"	physics	"HARRIS, Samuel"

Table 5.1: Example of contextual entities

the "HARRIS, Sam" appellation (denoted $\mathbb{O}_s = \{Nc_3, Nc_4, Nc_5, Nc_6, Nc_7, Nc_8\}$) and "PLATON" appellation ($\mathbb{O}_p = \{Nc_1, Nc_2\}$). This Table presents the id, the title, the publication date ("date"), the publication domain ("domain") as a keyword, and the appellation of the contributor C ("appellation") for each contextual entity. The human partition or expert-validated partitions for the two Sudoc subsets are respectively $Ph_s = \{\{Nc_5\}, \{Nc_3, Nc_4\}, \{Nc_7\}, \{Nc_6\}, \{Nc_8\}\}$ and $Ph_p = \{\{Nc_1, Nc_2\}\}$. We calculated if the human partitions are best partitions for the two separate Sudoc subsets and the union of them according to the criterion set $\mathbb{C} = \{appellation, domain, date, title\}$. Those criteria have been detailed in Section 4.2.2 page 50.

Example 39 (Sudoc subset related to "HARRIS, Sam" and global semantics)**.** *Let us apply global semantics on $\mathbb{O}_s = \{Nc_3, Nc_4, Nc_5, Nc_6, Nc_7, Nc_8\}$. This object set is not coherent with respect to our criteria. The Ph_s value is such that:*

- *$v(Ph_s, domain) = (always, -)$ (it exists an edge labelled $+++++$ that is not satisfied, between Nc_3 and Nc_4),*
- *$v(Ph_s, date) = (always, --)$ (it exists an edge labelled $-$ that is not satisfied, between Nc_6 and Nc_7).*

Ph_s has a best partition value on $\mathbb{O}_s$. However, partitions $P'_s = \{\{Nc_5, Nc_3, Nc_4\}, \{Nc_7\}, \{Nc_6\}, \{Nc_8\}\}$ with $(+, -)$ value for domain criterion and $(always, never)$ value for date criterion is also a best partition. The plurality of best partition values comes from incoherence between the date and domain criteria.

Example 40 (Sudoc subset related to "PLATON" and global semantics)**.** *Let us now apply global semantics on $\mathbb{O}_p = \{Nc_1, Nc_2\}$.The expert-validated partition is $Ph_p = \{\{Nc_1, Nc_2\}\}$. There is an incoherence between date and title criteria. Ph_p value is such that:*

- *$v(Ph_p, date) = (always, never)$. (it exists an edge labelled $--$ that is not satisfied, between Nc_1 and Nc_2).*

Ph_p is the only possible best partition on $\mathbb{O}_p$ because $\{\{Nc_1\}, \{Nc_2\}\}$ is not valid with respect to title criterion.

Let us now illustrate how global semantics will affect the whole set of objects by computing the best global partition on the union of contextual entities of the two Sudoc subsets.

Example 41 (Union of Sudoc subsets related to "HARRIS, Sam" and "PLATON" and global semantics)**.** *We now apply global semantics on all our selected contextual authorities:* $\mathbb{O} = \mathbb{O}_p \cup \mathbb{O}_s$*. The expert-validated partition is* $Ph_{ps} = \{\{Nc_1, Nc_2\}, \{Nc_3, Nc_4\}, \{Nc_5\}, \{Nc_7\}, \{Nc_6\}, \{Nc_8\}\}$*. We also encounter incoherences and this partition has the worst of* Ph_p *and* Ph_s *values for each criterion, in particular:*

- $v(Ph_{ps}, domains) = (always, -)$ *(it exists an edge labelled* $+++++$ *that is not satisfied, between* Nc_3 *and* Nc_4*, as in Example 39),*
- $v(Ph_{ps}, date) = (always, never)$ *(it exists an edge labelled* $--$ *that is not satisfied, between* Nc_1 *and* Nc_2*, as in Example 40).*

Ph_{ps} *has not the best partition value because we could improve partition value for domain criterion. This does not affect the date criterion value because it is already as bad as possible. For example, partition* $P'_{ps} = \{\{Nc_1, Nc_2\}, \{Nc_3, Nc_4, Nc_5\}, \{Nc_6, Nc_7\}, \{Nc_8\}\}$ *has a best value.*

A way to fix this problem is to propose the local semantics detailed in the next Section 5.3, which adds a notion of locality to global semantics.

5.3 Local semantics

In this Section, we will define how local semantics evaluates and compares partitions. In order to do that, we need to explain notions about incoherences in detail. This will be done in Section 5.3.1. Once it is done, we will present the algorithm that finds all best partition values on a criteria graph according to local semantics in Section 5.3.2. This algorithm extends Algorithm 3 (which finds best partition values according to global semantics and several criteria) by adding a locality notion.

5.3.1 Incoherences in a criteria graph

Local semantics do not consider incoherence for the whole treated object set (denoted $\mathbb{O}$) but only for objects that cause incoherences. As explained page 61 and Example 26, a pair of objects that causes incoherence is a pair of objects that must be put in the same class according to some criteria and kept separated according to others.

In order to understand how local semantics treat incoherences, we require to define the notions of incoherent parts and incoherent subsets of a criteria graph. Intuitively, an incoherent subset of an object set $\mathbb{O}$ according to a criterion set $\mathbb{C}$ is a minimal subset of $\mathbb{O}$ such that there is an incoherence in it and objects in this incoherent subset are not linked to objects not in it by *always* or closeness

comparisons values according to the criterion set $\mathbb{C}$. Incoherent parts are a way to divide a criteria graph into independent parts that could be evaluated separately because they concern objects that have nothing to do with objects in a distinct independent part according to the considered criterion set. In order to formally define independent parts, let us define minimal independent subsets. Intuitively, a minimal independent subset is a connected component of $\mathbb{G}_{\mathbb{C}}$ such that we consider the edges labelled by a closeness or *always* comparison value and not the other edges (labelled by a farness, *never* or *neutral* comparison value).

Definition 23 (Minimal independent subsets)**.** *Let $\mathbb{O}$ be an object set and $\mathbb{C}$, a criterion set on $\mathbb{O}$. A **minimal independent subset** $\mathbb{I}$ of $\mathbb{O}$ according to $\mathbb{C}$ is a subset of $\mathbb{O}$ such that:*

- *there are no always or closeness comparison values between an object of $\mathbb{O} \setminus \mathbb{I}$ and an object in $\mathbb{I}$ according to a criterion of $\mathbb{C}$, and*
- *there is no subset of $\mathbb{I}$ that is a minimal incoherent subset of $\mathbb{O}$.*

Definition 24 (Minimal incoherent subsets)**.** *Let $\mathbb{O}$ be an object set and $\mathbb{C}$, a criterion set on $\mathbb{O}$. A **minimal incoherent subset** $\mathbb{I}$ of $\mathbb{O}$ according to $\mathbb{C}$ is an independent subset (as defined in Definition 23) of $\mathbb{O}$ that contains a pair of objects that causes incoherences.*

Example 42 (Independent and incoherent subset)**.** *Let us consider the $\mathbb{G}^u_{\{C_1,C_2\}}$ criteria graph represented on Figure a) 5.4. There are three minimal incoherent subsets in $\{O\}_u = \{a,b,c,d,e,f,g\}$ according to $\{C_1,C_2\}$: $\{a,b,c,d,e\}$, $\{f,g\}$ and $\{h,i\}$. Two of them are also incoherent subsets: $\{a,b,c,d,e\}$ and $\{f,g\}$.*

Independent parts are deduced from a minimal independent subset $\mathbb{I}$. An independent part deduced from $\mathbb{I}$ contains all $\mathbb{O}$ objects but does not considers comparison values for every couple of objects that are not both occurring in $\mathbb{I}$. If $\mathbb{I}$ is an incoherent subset, the part deduced is also an incoherent part.

Definition 25 (Independent parts)**.** *Let $\mathbb{G}_{\mathbb{C}}$ be the criterion graph that represents the partitioning problem of the object set $\mathbb{O}$ according to the criterion set $\mathbb{C}$.*

The independent part $IndP$ of $\mathbb{G}_{\mathbb{C}}$ according to a minimal independent subset $\mathbb{I}$ is a sub-graph of $\mathbb{G}_{\mathbb{C}}$ such that all edges that are not between two objects in $\mathbb{I}$ are removed.

The independent part of $\mathbb{G}_{\mathbb{C}}$ according to the minimal subset $\mathbb{I}$ is denoted $independentPart(\mathbb{G}_{\mathbb{C}}, \mathbb{I})$, and abusively $independentPart(\mathbb{I})$ when $\mathbb{G}_{\mathbb{C}}$ is obvious.

Definition 26 (Incoherent parts)**.** *Let $\mathbb{G}_{\mathbb{C}}$ be the criterion graph that represents the partitioning problem of the object set $\mathbb{O}$ according to the criterion set $\mathbb{C}$.*

An incoherent part IP of $\mathbb{G}_{\mathbb{C}}$ is an independent part (please see Definition 25) of $\mathbb{G}_{\mathbb{C}}$ according to a minimal incoherent subset $\mathbb{I}$.

The incoherent part of $\mathbb{G}_{\mathbb{C}}$ according to the minimal subset $\mathbb{I}$ is denoted $incoherentPart(\mathbb{G}_{\mathbb{C}}, \mathbb{I})$, and abusively $incoherentPart(\mathbb{I})$ when $\mathbb{G}_{\mathbb{C}}$ is obvious.

Example 43 (Independent and incoherent parts)**.** *In Example 42, we detailed the minimal independent subsets of the* $\mathbb{G}^u_{\{C_1,C_2\}}$ *criteria graph represented on Figure a) 5.4.*

The independent parts according to each of those subsets are represented on Figure 5.4: incoherentPart$(\{a,b,c,d,e\})$ *on Figure b) 5.4, incoherentPart*$(\{f,g\})$ *on Figure c) 5.4 and the independentPart*$(\{h,i\})$ *on Figure d) 5.4.*

Since two of the independent subsets were incoherent ($\{a,b,c,d\}$ *and* $\{f,g\}$*), the deduced independent parts are also incoherent parts.*

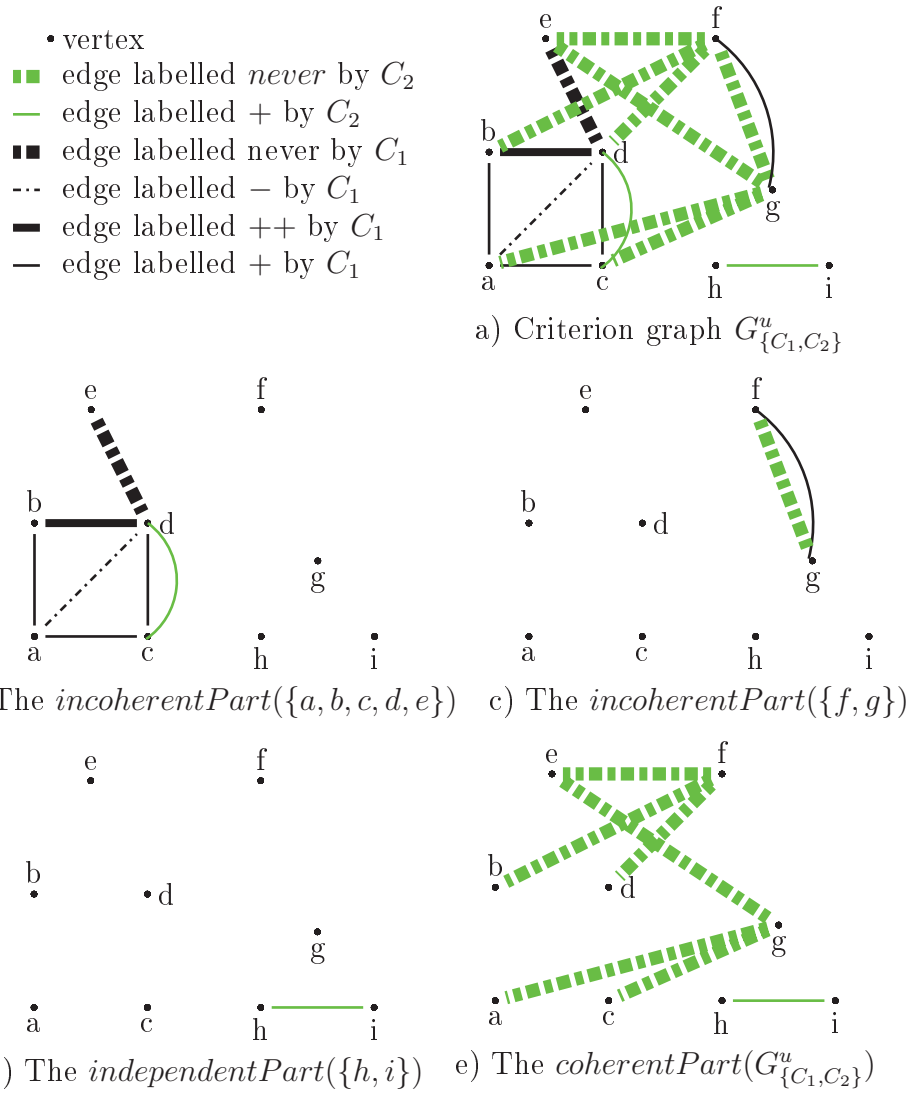

Figure 5.4: Parts of criteria graph $G^u_{\{C_1,C_2\}}$

The coherent part contains all $\mathbb{O}$ objects but does not consider the comparison

values for every pair of objects that are occurring in the same minimal incoherent subset.

Definition 27 (Coherent part). *Let $\mathbb{G}_{\mathbb{C}}$ be the criterion graph that represents the partitioning problem of the object set $\mathbb{O}$ according to the criterion set $\mathbb{C}$. Let us denote $\mathbb{IP}$ the set of all incoherent parts on $\mathbb{G}_{\mathbb{C}}$ (please see Definition 26).*

The coherent part of $\mathbb{G}_{\mathbb{C}}$ according to $\mathbb{C}$ is a sub-graph of $\mathbb{G}_{\mathbb{C}}$ such that all edges contained in a incoherent part in $\mathbb{IP}$ are removed. The coherent part of $\mathbb{G}_{\mathbb{C}}$ according to $\mathbb{C}$ is denoted $coherentPart(\mathbb{G}_{\mathbb{C}})$.

Example 44 (Coherent part). *In Example 43 we presented the incoherent parts of the $\mathbb{G}^u_{\{C_1,C_2\}}$ criteria graph represented on Figure 5.4. Let us represent the coherent part of $\mathbb{G}^u_{\{C_1,C_2\}}$, $coherentPart(\mathbb{G}^u_{\{C_1,C_2\}})$ on Figure e) 5.4.*

A partition on $\mathbb{O}$ is better than another partition if it has a best value for the coherent part and for each incoherent part. The values of each (in)coherent part are determined by global semantics.

Let us detail an Example in the aim of showing how local semantics is useful to add the locality notion to global semantics.

Example 45 (Local semantics interest). *In Examples 39, 40 and 41, we saw that global semantics can consider two partitions (P_s and P_p in this case) as best partitions on their respective object sets ($\mathbb{O}_s$ and $\mathbb{O}_p$) but the union of these best partitions is not a best partition on the union of the object set. In this case, it is a problem because objects of $\mathbb{O}_s$ have nothing to do with objects of $\mathbb{O}_p$, so analysing them together or separately must not change the results.*

The partition $P_p = \{\{Nc_1, Nc_2\}\}$ on $\mathbb{O}_p$ has the same value according to local semantics as according to global semantics (in Example 40) because $\mathbb{O}_p$ is composed of a single incoherent part ($\{Nc_1, Nc_2\}$).

The partition $P_s = \{\{Nc_3, Nc_4\}, \{Nc_5\}, \{, Nc_6, Nc_7\}, \{Nc_8\}\}$ has a best value according to global semantics (in Example 39) and according to global semantics. Indeed, the coherent part has a best value, as each incoherent part ($\{Nc_3, Nc_4, Nc_5\}$ and $\{, Nc_6, Nc_7\}$:

- *The P_s has a value for the $incoherentPart(\{Nc_3, Nc_4, Nc_5\})$ such that $v(Ph_s, domain) = (always, -)$. This value is a best one for $incoherentPart(\{Nc_3, Nc_4, Nc_5\})$ because of the incoherence between date and domain criteria for objects Nc_4 and Nc_5.*

- *The P_p has a value for the $incoherentPart(\{Nc_6, Nc_7\})$ such that $v(Ph_p, date) = (always, --)$. This value is a best one for $incoherentPart(\{Nc_6, Nc_7\})$ because of the incoherence between title and date criteria for objects Nc_6 and Nc_7.*

Contrary to global semantics, the partition $Ph_{ps} = P_p \cap P_s = \{\{Nc_1, Nc_2\}, \{Nc_3, Nc_4\}, \{Nc_5\}, \{Nc_7\}$ has a best value on the union on $\mathbb{O}_s \cap \mathbb{O}_p$. In this case,

there are three incoherent parts: incoherentPart($\{Nc_1, Nc_2\}$)*, incoherentPart*($\{Nc_3$, Nc_4, $Nc_5\}$) *and incoherentPart*($\{Nc_6, Nc_7\}$)*. Each of them has exactly the same value as previously when it was evaluated for the object sets* $\mathbb{O}_s$ *or* $\mathbb{O}_p$ *since in this semantics the incoherent subsets are considered independently from the others. The* Ph_{ps} *has also a best partition value for the coherent part, so* Ph_{ps} *has a best partition value for* $\mathbb{O}_s \cap \mathbb{O}_p$*.*

Dividing graph into independent parts (incoherent or coherent parts) allows the "best partition" function to be distributive over the union of independent object sets[3] according to a criterion set.

In other words, a partition $P_{i,j} = P_i \cup P_j$ on the object set $\mathbb{O}_{i,j} = \mathbb{O}_i \cup \mathbb{O}_j$ is a best partition on $\mathbb{O}_{i,j}$ with respect to the criterion set $\mathbb{C}$ if and only if P_i and P_j are respectively best partitions on $\mathbb{O}_i$ and $\mathbb{O}_j$ with respect to $\mathbb{C}$.

This is because the local semantics treats each independent parts and the coherent part apart. In this case, the incoherent parts of $\mathbb{O}_i$ are distinct from independent parts of $\mathbb{O}_j$ because $\mathbb{O}_i$ and $\mathbb{O}_j$ are disjoint. More of that, independent parts of $\mathbb{O}_i$ are independent from independent parts of $\mathbb{O}_j$ because $\mathbb{O}_i$ and $\mathbb{O}_j$, so the independent parts of $\mathbb{O}_{i,j}$ are exactly the union of independent part sets of $\mathbb{O}_i$ and $\mathbb{O}_j$. The coherent part, by definition, does not contain incoherences. Consequently, there is a single best value with respect to a criterion set, and best partitions with respect to the same criterion set have necessarily the same coherent part value. This explains why $P_{i,j}$ on the coherent part of $\mathbb{O}_{i,j}$ has the same value as P_i on the coherent part of $\mathbb{O}_i$ and as P_j on the coherent part of $\mathbb{O}_j$.

In this Section, we detailed how to divide a criteria graph in independent parts (coherent and incoherent parts), in the aim of evaluating them separately. We saw how local semantics use them. Let us present the algorithm used to find all best partition values according to local semantics in the following Section.

5.3.2 Finding best partitions for several criteria

To find all best partition values on a criteria graph according to local semantics, we first need to identify incoherent (and coherent) parts with a "finding connected components" algorithm (complexity $\mathcal{O}(m \log n)$ with n vertexes and m edges). Then, for the coherent part and each of the at most $n/2$ incoherent parts [4], we execute the algorithm of global semantics (of complexity $\mathcal{O}((k+1)^c * m \log n)$). The algorithm that finds best partition values according to local semantics and a criterion set is Algorithm 4 and is named *localAlgorithm*.

Complexity. The complexity in the worst case of *localAlgorithm* is: $\mathcal{O}(n * (k+1)^c * m \log n)$ (please see Algorithm 4).

[3] Two object sets $\mathbb{O}_i$ and $\mathbb{O}_j$ are independent from each other according to a criterion set $\mathbb{C}$ if and only if they are disjoint and there are no objects $o_i \in \mathbb{O}_i$, $o_j \in \mathbb{O}_j$ and a criterion $C \in \mathbb{C}$ such that $C(o_i, o_j) \in V^C_{close}$. This is very similar to Definition 23.

[4] since an incoherent part contains at least two edges between two vertexes

Algorithm 4 localAlgorithm

Require: $\mathbb{C}$, a criterion set on an object set $\mathbb{O}$; $\mathbb{G}_{\mathbb{C}}$ criteria graph of $\mathbb{C}$
Ensure: set of best partition values with respect to $\mathbb{C}$ on $\mathbb{O}$.

1: best partition value set $bestV = \{\}$;
2: Partition $P_a = ref(\mathbb{VP})$;
3: set of graphs: $Gparts = \{\}$;
4: **for** each incoherent class $\mathbb{I} \in P_a$ **do**
5: add $incoherentPart(\mathbb{G}_{\mathbb{C}}, \mathbb{I})$ to $Gparts$;
6: **end for**
7: add $coherentPart(\mathbb{G}_{\mathbb{C}}, Gparts)$ to $Gparts$;
8: apply Algorithm 3 on each graph in $Gparts$;
9: $bestV = \{$ best partition for $\mathbb{G}_{\mathbb{C}}$: best partition for each graph in $Gparts\}$;
10: **return** $bestV$;

Let us expose some details about the implementation of algorithms in the following Section before concluding this Chapter.

5.4 About the implementation of partitioning semantics algorithms

Let us expose in this Section some precisions about implementation.

As explained in Section 4.1.1 page 39, the Sudoc is initially in a Marc version. In the context of the SudocAd project [21], the Sudoc has been translated into an RDF(S) version. The n-triples version is based on the RDF(S) version and is the one used in this thesis. Contextual entities and authority notices related to a Sudoc subset are exported into a file in the n-triples version. There is a file per Sudoc subset.

The file containing the n-triples version of the Sudoc subset we are interested into is read and queried by Alaska [25] in order to identify the contextual entities and their attributes. Once it is done, the criteria graph is filled up.

The criteria graph is coded by a set of lists of edges: there is a list of edges per comparison value of each criterion. Partitions are represented by a table such that the indices correspond to the objects to partition and the values to the classes affected to the objects. Algorithms are implemented in Java 1.6.

Conclusion. The proposed partitioning semantics allows to find that objects are close together with respect to a criterion set. This can be used on Sudoc subsets in order to detect erroneous links by comparing the initial partition value to the best partition values, as it has been explained in Section 3.2.3 page 37. However, to detect erroneous links is not enough to repair them. This is why repair algorithms are proposed in the following Chapter.

Chapter 6

Repair algorithms

In Chapter 5, we presented two partitioning semantics that return a partition value for any partition. This allows us to compare partitions amongst them and especially with respect to special partitions: initial and human partitions (defined in Section 3.2.3 page 37). Unfortunately, these semantics do not allow us to improve the value of a partition. This is the aim of repair algorithms that are presented in this Chapter. We will start by explaining what we expect of a repair algorithm in Section 6.1 before detailing the proposed repair algorithms in Sections 6.2, 6.3, 6.4 and 6.5. More precisely, Section 6.2 proposes a naive algorithm; Section 6.3 an algorithm that refines the first naive algorithm in certain cases and Section 6.4, an algorithm that takes care of the source notion (guaranteed good links) unlike to the previous algorithm. Finally, in Section 6.5, we will add a last step to the previous algorithm in the aim of improving its result.

Each of those four algorithms are theoretically evaluated in their own Section.

6.1 Introduction, data, why, definition, hypothesis

Let P be a partition on an object set $\mathbb{O}$, $\mathbb{C}$ a criterion set that compares the objects of $\mathbb{O}$, and v, a best partition value on $\mathbb{O}$ with respect to $\mathbb{C}$.

If the value of P on $\mathbb{O}$ with respect to $\mathbb{C}$ is worse than, or incomparable to v, the repair aims to transform the P partition (by division or merge of classes, please see Definition 28) into a P' partition that has v as partition value, and doing so while making the smallest possible number of atomic transformations (i.e. merges and divisions) and taking into account sources. Sources are explained page 42. Intuitively, the sources of an authority notice are the bibliographic notices linked to it such that we assume that these links are good. In this Chapter, a source is an object of $\mathbb{O}$ that has a particular meaning. It represents the importance of the relation between an authority notice and the bibliographic notice for which it was created (please see Section 6.1.1). Some algorithms will also use composed transformations: replacement and newplacement of vertexes (please see Definition 29).

When a partition P' has no more unsatisfied edges[1] that forbid P' to have v as partition value, P' has v as partition value. Those particular unsatisfied edges are named unaccepted edges. The goal of repair algorithms is to transform a given partition into a new partition that has a particular partition value. The repair algorithms focus on those unaccepted edges.

First, we will explain the source notion (Section 6.1.1), then, the distinct types of transformations (Section 6.1.2) and finally the notions around interesting edges in the aim of fixing partitions (Section 6.1.3).

6.1.1 Sources

In Chapter 4, sources of an authority notice are the bibliographic notices linked to it such that this link is assumed to be good. Intuitively, in this Chapter, sources are contextual entities that represent such a link. Let us explain what is a source in this Chapter.

Let Pi be the initial partition on an authority notice set $\mathbb{O}$. A contextual entity of $\mathbb{O}$ is a **source** if and only if the represented contextual entity was created with an authority notice Na_i and its own source-bibliographical notice Nb_j. This means that Na_i was created to represent a contributor of Nb_j (please see page 42). As a consequence, we make the hypothesis that there is at most a single source for each class of the initial partition[2] Pi on $\mathbb{G}_{\mathbb{C}}$, and we denote $source(x, Pi)$ the source in the class $class(x, Pi)$ for each vertex x of $\mathbb{G}_{\mathbb{C}}$ with Pi, the initial partition.

However, sometimes there are several sources (contextual entities) for the same authority notice. Those contextual entities are assumed to represent good links pointing to the same authority notice, so they have to be in the same class of any partition that can be a best partition. We can guarantee that by adding the following criterion to the criterion set used to partition a contextual entity set.

- The *source* criterion is a closeness-criterion[3]. Let Nc_i and Nc_j be two contextual entities to compare. The *source* criterion returns an always value ($source(Nc_i, Nc_j) =$

[1] We remind ourselves that an unsatisfied edge of a partition P on a criteria graph $\mathbb{G}_{\mathbb{C}}$ is an edge (o_i, o_j) such that o_i and o_j

- are in a same class of P and there is a criterion $C \in \mathbb{C}$ such that the comparison value between o_i and o_j according to C is a farness or *never* comparison value, or
- are in distinct classes of P and there is a criterion $C \in \mathbb{C}$ such that the comparison value between o_i and o_j according to C is a closeness or *always* comparison value.

Please see Definition 12 page 62 for further details.

[2] We remind ourselves that an initial partition is a partition such that contextual entities are in a same class if and only if they represent links linked to a same authority notice. Please see Definition 1 page 37 for details.

[3] A closeness criterion is a criterion that gives closeness or *always* comparison values, as explained in Definition 4 page 49.

always) if and only if Nc_i and Nc_j are sources of the same authority notice. In the other cases, the *source* criterion returns *neutral* value.

With this criterion, all sources of a same authority notice will be in the same class of all valid[4] partitions because if not, the partition is not valid. Because each partition that has a best value is valid, the repair algorithms that guarantee to give as the result a partition with the desired (valid) partition value will only propose partition repairs such that all sources of the same authority notice are in the same class, even if we formally consider a single source by an authority notice.

6.1.2 Partition transformations

Let us present the transformations we use to make a partition P into a modified partition. There are two types: atomic transformations (Definition 28) and composed transformations (Definition 29).

Definition 28 (atomic transformation)**.** *Let P be a partition. An atomic transformation of P is:*

- *a **merge** of classes: two classes c_i, c_j of P are replaced by a single class $c_{ij} = c_i \cup c_j$ (a merge returns a new partition that is denoted $merge(c_i, c_j, P)$), or*
- *a **division** of a class: a class c of P is replaced by two new classes c_i, c_j such that $c_i \cup c_j = c$, $c_i \cap c_j = \{\}$ and $c_i, c_j \neq \{\}$ (a division returns a new partition that is denoted $division(c_i, P)$).*

Example 46 (Atomic transformation)**.** *Let P be the partition $\{\{a, b, c, d\}, \{e, f, g, h, i, j\}\}$.*

- $merge(\{a, b, c, d\}, \{e, f, g, h, i, j\}, P) = \{\{a, b, c, d, e, f, g, h, i, j\}\}$.
- $division(\{a, b\}, P) = \{\{a, b\}, \{c, d\}, \{e, f, g, h, i, j\}\}$.

Definition 29 (composed transformation)**.** *Let P be a partition. A composed transformation of P is:*

- *a **replacement** of vertexes between two classes: let c_i, c_j be two classes of P, and $c \subset c_i$. c_i and c_j are replaced by $c_i - c$ and $c_j \cup c$ (a replacement returns a new partition that is denoted $replacement(c, c_j, P)$);*
- *a **newplacement** of vertexes from two classes: let c_i, c_j be two classes of P, $c'_i \subset c_i$ and $c'_j \subset c_j$. c_i and c_j are replaced by $c_i - c'_i$, respectively $c_j - c'_j$*

[4]The validity of a partition is defined Definition 11 page 61. A partition which is not valid cannot be a best partition and is not even considered as a solution of the entity resolution problem. Intuitively, it is a partition such that there are two objects in the same class that have to be *never* together according a criterion, or on the contrary in distinct classes but they have to *always* be in the same class according to a criterion.

and $c_k = c'_i \cup c'_j$ (a newplacement returns a new partition that is denoted $newplacement(c_k, P)$).

A replacement counts as two atomic transformations (a division and a merge) and a newplacement counts as three atomic transformations (two divisions and a merge).

Example 47 (Complex transformation)**.** *Let P be the partition $\{\{a,b,c,d\},\{e,f,g,h,i,j\}\}$ defined Example 46.*

- $replacement(\{c\},\{e,f,g,h,i,j\},P) = \{\{a,b,d\},\{c,e,f,g,h,i,j\}\}$.
- $newplacement(\{c,e\},P) = \{\{a,b,d\},\{c,e\},\{f,g,h,i,j\}\}$.

The transformations in a partition correspond to link modifications in Sudoc. Intuitively, a merge corresponds to fuse two authority notices and to link their bibliography to the new authority notice. A division corresponds to creating a new authority notice and to linking it to a set of bibliographic notices originally linked to a same authority notice. A replacement corresponds to redirecting links from some bibliographic notices (originally linked to the same authority notice) to another existing authority notice. A newplacement corresponds to redirecting links from some bibliographic notices (originally linked to two distinct authority notices) to a new authority notice. The transformations indicate which bibliographic notices have to be linked to which authority notices, but do not indicate how to fuse two authority notices and their attributes or with which attributes to create a new authority notice when it is necessary.

In this Section we presented the transformations of partitions used by the proposed repair algorithm. Each transformation aims to reduce the number of edges that are unsatisfied, but not just any of them: the important ones for a partition having a chosen v partition value, i.e. the unaccepted edges.

6.1.3 Crucial and unaccepted edges

Intuitively, an unaccepted edge (Definition 12) for a partition P and a partition value v is an edge that is satisfied for any partition with v as partition value (called crucial edge, defined in Definitions 30 and 31) but is not satisfied in P. Those unaccepted edges are the edges that forbid P to have v as partition value.

Definition 30 (Crucial edge for a criterion)**.** *Let G_C be a criterion graph, $v(P,C) = (v_p, v_n)$, the partition value[5] of a partition P on G_C, and x, y two vertexes of G_C. An edge (x,y) is a crucial edge if $C(x,y) = v$ such that:*

[5] We recall ourselves that the value (v_p, v_n) of a partition P on a criterion graph G_C (defined in Definition 14 page 64) is such that:

- v_p is the weakest closeness or *always* value such that there is no unsatisfied edge in G_C according to P (an edge such that its vertexes are in distinct classes) labelled by a value

- $v \in V^C_{close} \cup \{always\}$ *and* $v \geq v_p$ *((x, y) is said a closeness crucial edge denoted CCE), or*

- $v \in V^C_{far} \cup \{never\}$ *and* $v \leq v_n$ *((x, y) is said a farness crucial edge denoted FCE).*

The set of all crucial edges of G_C for $v(P, C)$ is denoted $crucialEdges(v(P, C), G_C)$ (and can be abusively denoted $crucialEdges(v(P, C))$ when G_C is obvious).

The set of closeness crucial edges is denoted $closeCrucialEdges(v(P, C), G_C)$ (and can be abusively denoted $closeCrucialEdges(v(P, C))$ when G_C is obvious).

The set of farness crucial edges is denoted $farCrucialEdges(v(P, C), G_C)$ (and can be abusively denoted $farCrucialEdges(v(P, C))$ when G_C is obvious).

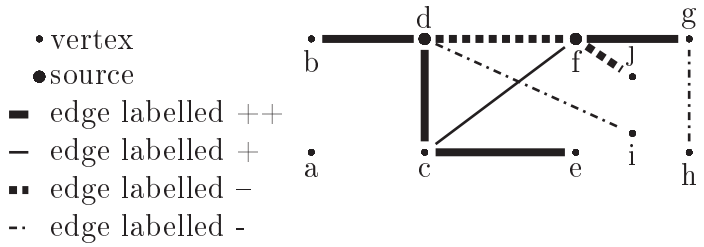

Figure 6.1: Criterion graph G_C

Example 48 (Crucial edge for a criterion graph)**.** *Let G_C, be the criterion graph represented on Figure 6.1, and $v = (++, -)$, a partition value.*

- $closeCrucialEdges(v) = \{(b, d), (d, c), (c, e), (f, g)\}$ *because*
  - $++ \in V^C_{close}$ *and* $C(b, d) = C(d, c) = C(c, e) = C(f, g) = ++$, *so* $\{(b, d), (d, c), (c, e), (f, g)\} \subseteq closeCrucialEdges(v)$;
  - $C(c, f) = +$, $+ \in V^C_{close}$ *but* $+ < ++$ *so* $C(b, d) \notin closeCrucialEdges(v)$.

- $farCrucialEdges(v) = \{(d, f), (g, h), (f, j), (i, d)\}$ *because*
  - $C(d, f), C(g, h), C(f, j), C(i, d) \in V^C_{far}$ *and*
  - $C(d, f), C(g, h), C(f, j), C(i, d) \leq -$

- $crucialEdges(v) = farCrucialEdges(v) \cup closeCrucialEdges(v) = \{(b, d), (d, c), (c, e), (f, g), (d, f), (g, h), (f, j), (i, d)\}$.

Crucial edges of G_C for the partition value $(++, -)$ are represented on Figure 6.2.

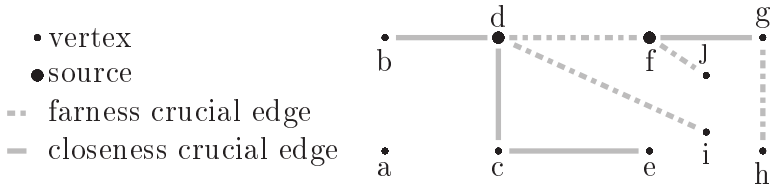

Figure 6.2: Crucial edges of G_C for the partition value $(++, -)$

Definition 31 (Crucial edge in a general case)**.** *Let $\mathbb{G}_{\mathbb{C}}$ be a criteria graph , and $v(P, \mathbb{C})$, the partition value*[6] *of a partition P on $\mathbb{G}_{\mathbb{C}}$. Let x, y be two vertexes of $\mathbb{G}_{\mathbb{C}}$. An edge (x, y) is:*

- *a crucial edge if and only if $\exists C \in \mathbb{C} | (x, y) \in crucialEdges(v(P, C), G_C)$ (please see Definition 30).*
- *a closeness crucial edge if and only if $\exists C \in \mathbb{C} | (x, y) \in closeCrucialEdges(v(P, C), G_C)$.*
- *a farness crucial edge if and only if $\exists C \in \mathbb{C} | (x, y) \in farCrucialEdges(v(P, C), G_C)$.*

The set of all crucial edges of $\mathbb{G}_{\mathbb{C}}$ for $v(P, \mathbb{C})$ is denoted $crucialEdges(v(P, \mathbb{C}), \mathbb{G}_{\mathbb{C}})$ (and can be abusively denoted $crucialEdges(v(P, \mathbb{C}))$ when $\mathbb{G}_{\mathbb{C}}$ is obvious).

The set of closeness crucial edges is denoted $closeCrucialEdges(v(P, \mathbb{C}), \mathbb{G}_{\mathbb{C}})$ (and can be abusively denoted $closeCrucialEdges(v(P, \mathbb{C}))$ when $\mathbb{G}_{\mathbb{C}}$ is obvious).

The set of farness crucial edges is denoted $farCrucialEdges(v(P, \mathbb{C}), \mathbb{G}_{\mathbb{C}})$ (and can be abusively denoted $farCrucialEdges(v(P, \mathbb{C}))$ when $\mathbb{G}_{\mathbb{C}}$ is obvious).

Definition 32 (Unaccepted edge)**.** *Let P, be a partition on a criterion graph $\mathbb{G}_{\mathbb{C}}$, and v, the partition value of a partition on $\mathbb{G}_{\mathbb{C}}$. An edge (x, y) is unaccepted for P with respect to v if and only if:*

- *(x, y) is crucial for the partition value v (please see Definition 31), and*
- *(x, y) is unsatisfied by P (please see Definition 12 page 62).*

We will denote $unacceptedEdges(v, P)$ the set of unaccepted (crucial) edges for P with respect to partition value v; $closeUnacceptedEdges(v, P)$ the set of unaccepted closeness (crucial) edges for P with respect to partition value v; and

---

stronger than v_p.

- v_n is the weakest farness or *never* value such that there is no unsatisfied edge in G_C according to P (an edge such that both vertexes are in the same class) labelled by a value stronger than v_n.

[6] We recall that in a general case, the partition value of a partition P on the criteria graph $\mathbb{G}_{\mathbb{C}}$ is the set of partition values of P on $\mathbb{G}_{\mathbb{C}}$ according to each criterion $C \in \mathbb{C}$. Please see Definition 16 page 66 for further details.

$farUnacceptedEdges(v, P)$ the set of unaccepted farness (crucial) edges for P with respect to partition value v.

Example 49 (Unaccepted edge)**.** *Let G_C, be the criterion graph and $v = (++, -)$ the partition value of Example 48. Let $\mathbb{G}_\mathbb{C}$ be the criteria graph such that $\mathbb{C} = \{C\}$ and $\mathbb{G}_\mathbb{C}$ vertexes are G_C vertexes. So, crucial edges of $\mathbb{G}_\mathbb{C}$ are exactly the crucial edges of G_C according to Definition 31, and are represented on Figure 6.2.*

Let us define the initial partition $P = \{\{a, b, c, d\}, \{e, f, g, h, j, i\}\}$ on $\mathbb{G}_\mathbb{C}$, represented on Figure 6.3 with the crucial edges on $\mathbb{G}_\mathbb{C}$. The crucial edges for v ($crucialEdges(v) = \{(b, d), (d, c), (c, e), (f, g), (d, f), (g, h), (f, j), (i, d)\}$ according to Example 48) which are unsatisfied by P are unaccepted by P with respect to v. So, $unacceptedEdges(v, P) = \{(c, e), (g, h), (f, j)\}$. (c, e) is a closeness unaccepted edge and (g, h), (f, j) are farness unaccepted edges.

The following Property asserts that partitions with no unaccepted edges according to a partition value v have a partition value better or equal to v. Intuitively, it is because a partition value depends on the edges with the most intense label (comparison value) which are unsatisfied. Removing them all improves the partition value.

Property 1 (Partitions with no unaccepted edge according to v)**.** *Let $\mathbb{G}_\mathbb{C}$ be a criterion graph, P a partition on $\mathbb{G}_\mathbb{C}$ and v a partition value of $\mathbb{G}_\mathbb{C}$. P has no unaccepted edge according to v, if and only if P has at least v as a partition value, or a better partition value.*

Proof 1 (Partitions with no unaccepted edge according to v)**.** *Let us consider a partition P on a criteria graph $\mathbb{G}_\mathbb{C}$ and a partition value v on $\mathbb{G}_\mathbb{C}$. Let us consider the set of unaccepted edge of P according to v: $unacceptedEdges(v, P)$.*

If there are unaccepted edges for P according to v ($unacceptedEdges(v, P) \neq \{\}$), then there is at least an unaccepted edge (x, y) according to P and v such that x and y are $\mathbb{G}_\mathbb{C}$ vertexes. An unaccepted edge can be a closeness or a farness unaccepted edge.

- *if (x, y) is a CUE, then:*
  - *x and y are in distinct classes of P ($class(x, P) \neq class(y, P)$), and*
  - *there is a criterion $C \in \mathbb{C}$ such that $C(x, y) = v'_p$ with $v'_p \geq v_p$, the inter value of v according to C.*

  Unfortunately, this means that the partition value of P according to C is $v(P, C) = (v'_p, v_n)$ at the best (with v_n, the intra value of v according to C), so $v(P, \mathbb{C})$ cannot be better than or equal to v if there are closeness unaccepted edges for P according to v.
- *if (x, y) is a FUE, then:*

- *x and y are in the same class of P ($class(x, P) = class(y, P)$), and*
- *there is a criterion $C \in \mathbb{C}$ such that $C(x, y) = v'_n$ with $v'_n \geq v_n$, the intra value of v according to C.*

Unfortunately, this means that the partition value of P according to C is $v(P, C) = (v_p, v'_n)$ at the best (with v_p, the inter value of v according to C), so $v(P, \mathbb{C})$ cannot be better than or equal to v if there are farness unaccepted edges for P according to v.

$v(P, \mathbb{C})$ cannot be better than or equal to v if there are unaccepted edges for P according to v. No unaccepted edges for P according to v implies that $v(P, \mathbb{C})$ is better than or equal to v because of the way unaccepted edges are defined.

A reference partition P_{ref} that has v as partition value is a partition, that, as any partition with v as partition value, does not have unaccepted edges according to v. Unaccepted edges are crucial unsatisfied edges; as a consequence, crucial edges are satisfied. This implies the following Property.

Property 2 (CCEs, FCEs and P_{ref})**.** *Let v be a best partition value on a graph $\mathbb{G}_\mathbb{C}$, $P_{ref} = Pref(v, \mathbb{G}_\mathbb{C})$ the unique reference partition on $\mathbb{G}_\mathbb{C}$ that has v as partition value, and (x, y) a crucial edge of $\mathbb{G}_\mathbb{C}$ with respect to the v value.*

- *If (x, y) is a closeness crucial edge, then vertexes x and y are in the same class of the reference partition P_{ref}: $class(x, P_{ref}) = class(y, P_{ref})$.*
- *If (x, y) is a farness crucial edge, then vertexes x and y are in distinct classes of the reference partition P_{ref}: $class(x, P_{ref}) \neq class(y, P_{ref})$.*

Proof 2 (CCEs, FCEs and P_{ref})**.** *Let P_{ref} be the reference partition on a criteria graph $\mathbb{G}_\mathbb{C}$ such that $v(P_{ref}, \mathbb{C}) = v$. Since P_{ref} has v as partition value, there are no crucial unaccepted edges in $\mathbb{G}_\mathbb{C}$ according to P and v (please see Property 1), which means that all crucial edges according to v are satisfied by P (according to in Definition 32). A crucial edge (x, y) can be a CCE or a FCE:*

- *if (x, y) is a closeness edge, then $class(x, P) = class(y, P)$ (according to satisfied edge, defined inDefinition 12 page 62);*
- *if (x, y) is a farness edge, then $class(x, P) \neq class(y, P)$ (according to satisfied edge, defined in Definition 12 page 62).*

The Property 2 is verified.

Let $\mathbb{G}_\mathbb{C}$ be a criteria graph and v, the value of a partition on $\mathbb{G}_\mathbb{C}$. Please note that if v is a best possible partition value on $\mathbb{G}_\mathbb{C}$, it means that there is at least a partition on $\mathbb{G}_\mathbb{C}$ with no unaccepted edges with respect to v. Those partitions have

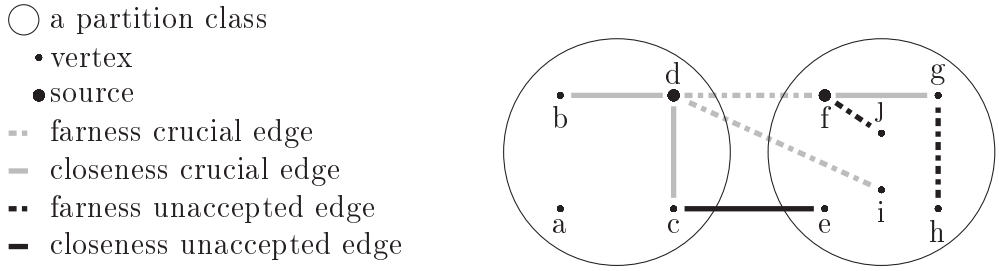

Figure 6.3: Unaccepted edges for the partition P on $\mathbb{G}_{\mathbb{C}}$ with respect to $(++, -)$

v as partition value. In particular, the reference partition[7] which has v as partition value has no unaccepted edges with respect to v.

In this Section, we detailed central concerns for a good repair algorithm: it should return a partition with the desired partition value (i.e. without unaccepted edges according to this desired partition value), go from the original partition to the resulting partition with the least possible number of atomic transformations between them, and take care of not separating vertexes from their sources when it is avoidable.

First, we will propose a naive algorithm for repairs based only on atomic repairs in Section 6.2. Then, we will propose an improved algorithm on Section 6.3 that uses composed transformations in the aim of refining the first naive algorithm in some cases but this increases the complexity ($\mathcal{O}(m * n)$ to $\mathcal{O}(m * (m + n))$ with n vertexes and m edges). In Section 6.4, the source repair algorithm will finally take care of the source notion, unlike the previous algorithm. Finally, in Section 6.5, we will add a last step in the algorithm in the aim of correcting the divisions of classes that could have been wiser.

6.2 Naive repair algorithm

Let P be a partition on a criterion graph $\mathbb{G}_{\mathbb{C}}$, and v a best possible partition value on $\mathbb{G}_{\mathbb{C}}$. In the reminder of this Chapter, we will only be interested in crucial edges.

We will denote $Pref(v, \mathbb{G}_{\mathbb{C}})$ the (unique) reference partition on $\mathbb{G}_{\mathbb{C}}$ that has v as partition value.

The naive algorithm (Algorithm 5) is intuitive. It has two steps: first, we would like to divide classes c of P in several classes such that there would be no more farness unaccepted edges, and no new closeness unaccepted edges. Second, once it is done, we merge classes in the aim of having no more closeness unaccepted edges, like in Algorithm 5. The resulting partition is P'. At the beginning, P' is a copy of

[7] We recall that a reference partition that has v as partition value is the most refined partition (Definition 18 page 68) with v as partition value. Please see Definition 22 page 72 for further details.

P. We note that there is at least a partition that has the desired partition value: P_{ref}.

Division step: (lines 2 to 4 from Algorithm 5). This step aims to have no more farness unaccepted edge, and no new closeness unaccepted edge with respect to the value v in the partition P'. We consider farness unaccepted edges (FUE) of P' one by one. For each FUE (x, y), the vertex x is put in a new partition class $class(x, P')$[8] distinct of $class(y, P')$. By doing this, the FUE (x, y) is no longer unaccepted because x and y are in distinct classes of P'. However, just separating x from $class(y, P') - \{x\}$ could create new closeness unaccepted edges (CUE) if there is a closeness crucial edge (CCE) (x, y') such that $y' \in class(y, P')$. In fact, (x, y') wasn't unaccepted when x was also in $class(y, P')$. In the aim of preventing new CUE in P', each vertex y' such that there is a CCE (x', y') with $x' \in class(x, P')$ and $y' \in class(y, P')$ is put in class $class(x, P')$ instead of $class(y, P')$ until there are no more vertexes $y' \in class(y, P')$ with this property. Those y' vertexes are all in the class $class(x, P_{ref})$ because (x, y') is a CCE and vertexes of a CCE according to partition value v are in the same class of P_{ref} (Property 2). This implies that $class(x, P')$ is a subset of $class(x, P_{ref})$, which does not contain farness crucial edges (FCE) with respect to v (Property 2), so $class(x, P')$ does not contain farness crucial edges. This allows the "division step" of the algorithm to end when all FUEs of P are treated.

We denote $around(x, P', P_{ref})$ the union of x to the y' vertexes which was in the same class of x in P' and are in the same class of x in any partition with v as partition value. They are defined in Definition 33.

Once it is done, there are no more FUEs in P', and no new CUEs in P'. We take care of CUEs in P' in the merge step of the naive repair algorithm (Algorithm 5).

The two following Properties 3 and 4 are used in the division step of Algorithm 5.

The Property 3 asserts that vertexes that are into the same class of the reference partition with v as partition value are into the same class of any partition that has also v as partition value. Intuitively, it is because the reference partition that has v as partition value is the most refined partition of the partitions that share the same partition value v. This implies that classes of the reference partition are always included in the classes of partitions with the same partition value v.

Property 3 (About vertexes of a same partition reference class)**.** *Let v be a best partition value on a graph $\mathbb{G}_{\mathbb{C}}$, $P_{ref} = Pref(v, \mathbb{G}_{\mathbb{C}})$, the unique reference partition on $\mathbb{G}_{\mathbb{C}}$ that has v as partition value, and x, y, two vertexes such that they are in the*

[8] We remind ourselves that the partition P class that contains x is denoted $class(x, P)$, as defined in Definition 6 page 58.

same class of P_{ref} ($class(x, P_{ref}) = class(y, P_{ref})$). Then in any partition on $\mathbb{G}_\mathbb{C}$ that has v as partition value, x and y are in the same class.

Proof 3 (About vertexes of a same partition reference class)**.** *Let $\mathbb{G}_\mathbb{C}$ be a criteria graph, v, a partition value on $\mathbb{G}_\mathbb{C}$ and P_{ref}, the reference partition on $\mathbb{G}_\mathbb{C}$ such that $v(P_{ref}, \mathbb{G}_\mathbb{C}) = v$.*

By definition, a reference partition is the most refined partition with its partition value as partition value (please see Definition 22 page 72), so P_{ref} is the most refined partition on $\mathbb{G}_\mathbb{C}$ that has v as partition value.

Let P be a partition on $\mathbb{G}_\mathbb{C}$ such that $v(P, \mathbb{C}) = v$ and x, y two vertexes of $\mathbb{G}_\mathbb{C}$ such that they are in the same class of P but not in the same class of P_{ref}.

This implies that P_{ref} is not more refined than P, so P_{ref} is not the most refined partition on $\mathbb{G}_\mathbb{C}$ with v as partition value: P_{ref} is not a reference partition, or P has not v as partition value.

As a consequence, the Property 3 is verified.

The Property 4 allows us to divide the classes of a partition P without creating new unaccepted edges with respect to a partition value v in the resulted partition. It is intuitively based on the idea that dividing a class of P without separating the vertexes of a same class in the reference partition P_{ref} which has v as partition value cannot add unaccepted edges in the resulted partition P' because:

- Separate vertexes cannot create FUEs, because a FUE is a FCE such that its vertexes are both inside the same class.
- For each CCE, its vertexes are in the same class of the reference partition, so dividing a class of P without separating the vertexes of a class of P_{ref} cannot add CUEs.

The Property 4 uses the following Definition which defines which vertexes have to stay together during the division of a partition class.

Definition 33 (The smallest subset around a vertex)**.** *Let P be a partition on graph $\mathbb{G}_\mathbb{C}$, x a vertex of $\mathbb{G}_\mathbb{C}$ and P_{ref} a best reference partition value such that $v(P_{ref}, \mathbb{G}_\mathbb{C}) = v$. The smallest subset around x in P according to v is $class(x, P) \cap class(x, P_{ref})$ and is denoted $around(x, P, P_{ref})$.*

Property 4 (Division with a smallest subset around x)**.** *Let P be a partition on graph $\mathbb{G}_\mathbb{C}$, x a vertex of $\mathbb{G}_\mathbb{C}$ and P_{ref} a best reference partition value such that $v(P_{ref}, \mathbb{G}_\mathbb{C}) = v$. Every unaccepted edge (y, w) in $division(around(x, P, P_{ref}), P)$ with respect to v is also unaccepted in P with respect to v partition value.*

Proof 4 (Division with a smallest subset around x)**.** *Let P be a partition on graph $\mathbb{G}_\mathbb{C}$, x a vertex of $\mathbb{G}_\mathbb{C}$ and P_{ref} a best reference partition value such that $v(P_{ref}, \mathbb{G}_\mathbb{C}) = v$, c_x the class of P containing x ($c_x = class(x, P)$) and c'_x the smallest subset around x of P denoted $c'_x = around(x, P, P_{ref})$ (Definition 33).*

Let P' be the partition such that $P' = division(around(x, P, P_{ref}), P)$. The class c_x of P has been replaced in P' by classes c'_x and $c_x - c'_x$ such that $c_x = c'_x \cup (c_x - c'_x)$. So, if a crucial edge according to v denoted (y, z) is unaccepted in P' but not in P, this edge must be:

1. *a closeness crucial edge and not a farness crucial edge (there are no two vertexes i,j such that $class(i, P)! = class(j, P)$ but not $class(i, P') = class(j, P')$;*

2. *such that $y \in c'_x$ and $z \in c_x - c'_x$ (or $z \in c'_x$ and $y \in c_x - c'_x$: we will focus on the first case only, because the second one is symmetric).*

If (z, y) is a closeness crucial edge according to v partition value, then $class(z, P_{ref}) = class(y, P_{ref})$ (Property 2). If $y \in c'_x$ and $z \in c_x - c'_x$, then $z, y \in c_x$ because $c_x = c'_x \cup (c_x - c'_x)$. $y \in c_x$ implies $y \in class(x, P_{ref})$ because $y \in c'_x$ and $c'_x = around(x, P, P_{ref}) = class(x, P) \cap class(x, P_{ref})$.

However, $z \in c_x - c'_x$ implies $z \notin class(x, P_{ref})$ because $c_x - c'_x = c_x - (class(x, P) \cap class(x, P_{ref}))$, so $c_x - c'_x = c_x - (c_x \cap class(x, P_{ref}))$ and $z \in c_x$.

By definition, there are no vertexes in two distinct classes of a same partition. However, $y \in class(x, P_{ref})$ and $z \notin class(x, P_{ref})$ but $class(z, P_{ref}) = class(y, P_{ref})$, which is impossible according to classes' definition. Considering that Property 4 is not true implies a non-sense, so, Property 4 is true and every unaccepted edge (y, w) in $division(around(x, P, P_{ref}), P)$ is also unaccepted in P according to the v partition value.

Merge step: The naive way to take care of CUEs in P' is to take each CUE (x, y) one by one, and, for each of them, merge $class(x, P')$ and $class(y, P')$ into one unique class. As we will see in Example 50, that could be useful for giving a partition that has v as partition value but that can also add farness unaccepted edges into the resulting partition, as shown in Example 51.

Example 50 (Naive algorithm)**.** *Let us take the crucial edges of the graph $\mathbb{G}_\mathbb{C}$ with respect to partition value $v = (++, -)$ described in Example 48. Let us define the partition P_a on $\mathbb{G}_\mathbb{C}$ such that $P_a = \{\{a, c, e, h\}, \{b, d\}, \{i, j, f, g\}\}$ and represented on Figure a) 6.4 with its unaccepted edges with respect to v.*

Let the partition P'_a be an exact copy of P_a, and $P_{ref} = \{\{a\}, \{b, c, d, e\}, \{f, g\}, \{h\}, \{i\}, \{j\}\}$ be the only reference partition such that $v(P_{ref}, \mathbb{G}_\mathbb{C}) = v$. The first step of Algorithm 5 is to identify FUE (here, (j, f)). The algorithm takes care of (j, f) and puts j in a new class with all the other vertexes belonging to the same class of P_{ref} and to $class(f, P'_a)$: the new class of j is $\{j\}$, and the new partition P'_a after $division(\{j\}, P'_a)$ is $P'_a = \{\{a, c, e, h\}, \{b, d\}, \{i, j, g\}, \{j\}\}$, represented on Figure b) 6.4.

There are no more FUE in P'.

The second step is to merge classes of vertexes of a same closeness unaccepted edge. There is a single CUE: (c, d). $class(c, P'_a) = \{a, c, e, h\}$ and $class(d, P'_a) =$

Algorithm 5 naiveRepairAlgorithm

Require: $\mathbb{G}_{\mathbb{C}} = (V, E)$, a criteria graph;
v, a possible partition value on $\mathbb{G}_{\mathbb{C}}$;
P, a partition on $\mathbb{G}_{\mathbb{C}}$;
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$;
Ensure: a partition P' that has v as partition value (or a better one)
1: Partition $P' = copy(P)$;
2: **while** $\exists(x, y)$, a farness unaccepted edge of P' for v **do**
3: $P' = division(around(x, P', P_{ref}), P')$;
4: **end while**
5: **while** $\exists(x, y)$, a closeness unaccepted edge of P' for v **do**
6: $P' = merge(class(x, P'), class(y, P'), P')$;
7: **end while**
8: **return** P';

$\{b, d\}$, *so the new and final* P'_a *partition, represented on Figure c) 6.4 is* $\{\{a, c, e, h, b, d\}$, $\{i, f, g\}$, $\{j\}\}$. *This partition has no more unaccepted edges and has* $v = (++, -)$ *as partition value.*

Example 51 (Counter example for naive algorithm)**.** *Let us take the unaccepted edges of the graph* $\mathbb{G}_{\mathbb{C}}$ *with respect to partition value* v *for the partition* P *described in Example 49 and represented on Figure 6.3.*

Let the partition P' *be an exact copy of* P, *and* $P_{ref} = \{\{a\}, \{b, c, d, e\}, \{f, g\}, \{h\}$, $\{i\}, \{j\}\}$ *be the only reference partition such that* $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$. *The first step of Algorithm 5 is to identify FUE (here,* (j, f) *and* (g, h)*). The algorithm takes care of* (j, f) *at first and puts* j *in a new class with all the other vertexes belonging to the same class of* P_{ref} *and to* $class(f, P')$*: the new class of* j *is* $\{j\}$, *and the new partition* P' *after* $division(\{j\}, P')$ *is represented on Figure a) 6.5.* (g, h) *is still a FUE so the algorithm puts* g *into a new class with all the other vertexes belonging to the same class of* P_{ref} *and to* $class(h, P')$*:* f. *The new class of* g *is* $\{g, f\}$, *and the new partition* $P' = \{\{a, b, c, d\}, \{e, h, i\}, \{f, g\}, \{j\}\}$ *after* $division(\{g, f\}, P')$ *is represented on Figure b) 6.5.*

There are no more FUEs in P'.

The second step is to merge classes of vertexes of a same closeness unaccepted edge. There is a single CUE: (c, e). $class(c, P') = \{a, b, c, d\}$ *and* $class(e, P') = \{e, h, i\}$, *so the new and final* P' *partition, represented on Figure c) 6.5 is* $\{\{a, b, c, d,$ $e, h, i\}$, $\{f, g\}, \{j\}\}$. *Unfortunately, there is a farness unaccepted edge,* (d, i), *that is not accepted and shows that the naive algorithm does not provide each time a modified partition with the desired value, or without unaccepted edges.*

Complexity and transformation number. Let us denote f the number of FUEs in the partition P on $\mathbb{G}_{\mathbb{C}}$ according to the partition value v. The division step

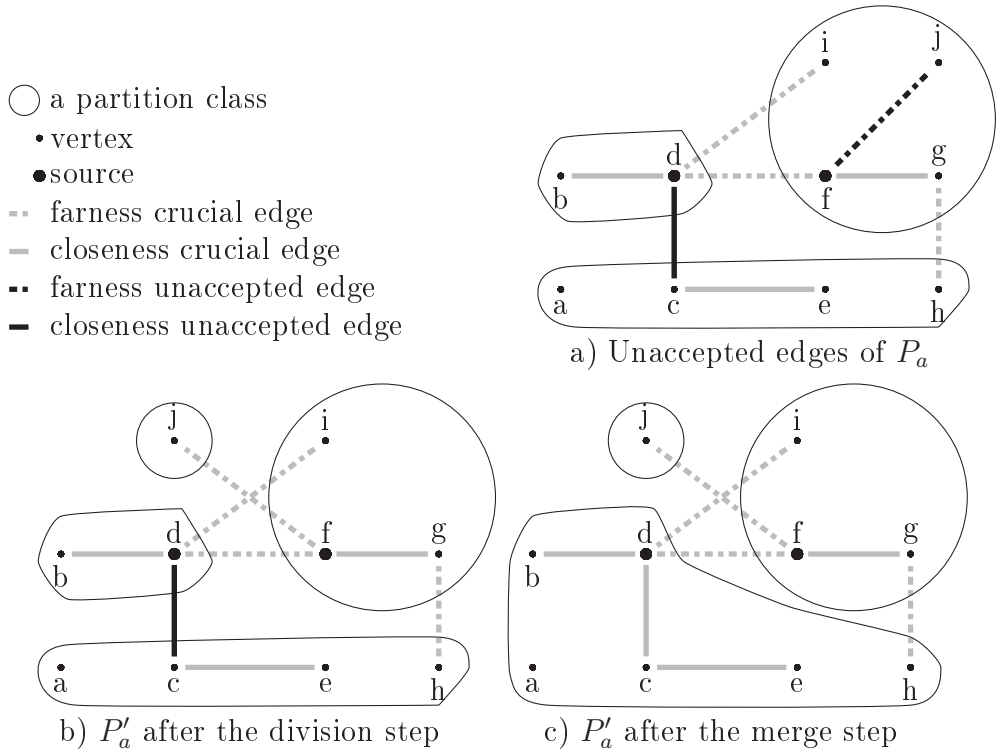

Figure 6.4: Modifications of P_a on $\mathbb{G}_{\mathbb{C}}$ with respect to $(++,-)$: example for the naive algorithm (5)

of Algorithm 5 does one division per FUE in the worst case, so it does f divisions (or atomic transformations) in the worst case. The complexity of the naive repair algorithm division step is $\mathcal{O}(m*n)$, with m edges (FUEs) to check in the worst case and n vertexes.

The complexity of the naive repair algorithm merge step is also $\mathcal{O}(m*n)$, with m edges (CUEs) to check in the worst case and n vertexes. Indeed, it does a fusion of classes per CUE in the worst case. So, the complexity of Algorithm 5 is $\mathcal{O}(m*n)$.

We saw in Example 51 that the naive repair algorithm (Algorithm 5) did not always give a partition free of unaccepted edges as the result. Let us correct this point with the intuitive repair algorithm.

6.3 Intuitive repair algorithm

The intuitive repair algorithm (Algorithm 6) proposes a smarter way to take care of CUEs after the division step of naive algorithm (Section 6.2) in the aim of not having new FUE after merges. Let us detail this merge step.

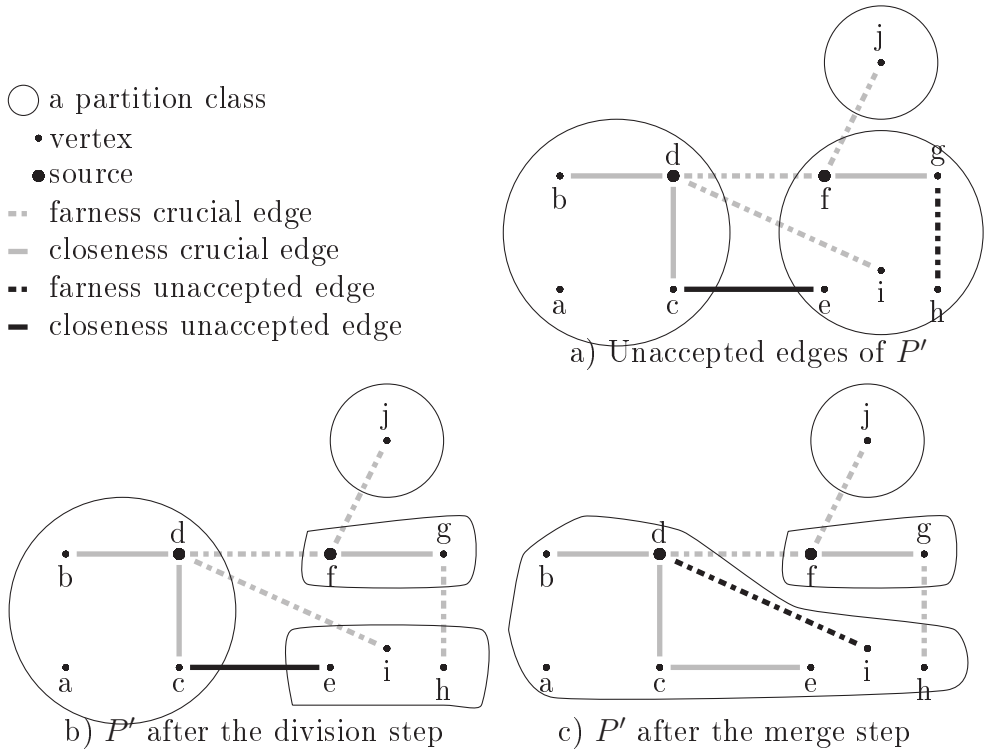

Figure 6.5: Modifications of P' on $\mathbb{G}_C$ with respect to $(++,-)$: counter example for the naive algorithm (Algorithm 5)

Merge step: (lines 5 to 26 from Algorithm 6). The intuitive repair algorithm, after the division step of the naive repair algorithm (Algorithm 5), takes CUEs of P' one by one. For each CUE (x, y), it checks if the merge of their respective classes ($class(x, P')$ and $class(y, P')$) adds some FUE in the resulted partition.

- If not, the algorithm merges those classes like the naive repair algorithm (Algorithm 5) does.

- If it is the case, like in a division, the algorithm tries to put x (and the smallest subset around x defined in Definition 33, in the aim of not adding a new CUE as for divisions of naive algorithm) into class $class(y, P')$, or in the same way, y into $class(x, P')$. For each, it checks if this does not create a new FUE in the resulting partition. If none of these possibilities is accepted, it creates a new class and puts into it x, y and vertexes around x and y, like in the naive algorithm's division step (in the aim of not creating a FUE in the resulting partition).

At the end of the process, there are no more FUEs in the partition because of the division step, and no more CUEs because of the merge step, so we obtain a partition

with the desired partition value v. Let us show how the algorithm handles CUEs with Example 52.

Algorithm 6 intuitiveRepairAlgorithm

Require: $\mathbb{G}_{\mathbb{C}} = (V, E)$, a criteria graph;
v, a possible partition value for $\mathbb{G}_{\mathbb{C}}$;
P, a partition on $\mathbb{G}_{\mathbb{C}}$;
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$.
Ensure: a partition P' that has v as partition value (or a better one)

1: Partition $P' = copy(P)$;
2: **while** $\exists (x, y) \in farUnacceptedEdge(v, P')$ **do**
3: $P' = division(around(x, P', P_{ref}), P')$;
4: **end while**
5: **while** $\exists (x, y) \in closeUnacceptedEdge(v, P')$ **do**
6: Class $c_x = class(x, P')$;
7: Class $c_y = class(y, P')$;
8: Partition $P_{test} = merge(c_x, c_y, P')$;
9: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | (z, w) \notin farUnacceptedEdge(v, P')$ **then**
10: $P' = P_{test}$;
11: **else**
12: Class $c'_x = around(x, P', P_{ref})$;
13: Class $c'_y = around(y, P', P_{ref})$;
14: $P_{test} = replacement(c'_x, c_y, P')$;
15: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | (z, w) \notin farUnacceptedEdge(v, P')$ **then**
16: $P' = P_{test}$;
17: **else**
18: $P_{test} = replacement(c'_y, c_x, P')$;
19: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | (z, w) \notin farUnacceptedEdge(v, P')$ **then**
20: $P' = P_{test}$;
21: **else**
22: $P' = newplacement(c'_x \cup c'_y, P')$;
23: **end if**
24: **end if**
25: **end if**
26: **end while**
27: **return** P';

Example 52 (Intuitive algorithm)**.** *Let us take the unaccepted edges of the graph $\mathbb{G}_{\mathbb{C}}$ with respect to partition value v for the partition P described in Example 49 and represented on Figure 6.3.*

The division step *of the intuitive repair algorithm (Algorithm 6) is exactly the same as the division step of the naive repair algorithm (Algorithm 5). So, after this step, the modified partition is* $P' = \{\{a, b, c, d\}, \{e, h, i\}, \{f, g\}, \{j\}\}$ *and is represented on Figure b) 6.5. There is no FUE in* P' *for the partition value* v.

The merge step *takes care of closeness unaccepted edges. In this case, there is a single CUE:* (c, e). $class(c, P') = \{a, b, c, d\}$ *and* $class(c, P') = \{e, h, i\}$.

Merging those classes like in the naive algorithm is not an option because the resulting partition $\{\{a, b, c, d, e, h, i\}, \{f, g\}, \{j\}\}$ *contains a FUE,* (c, i).

So, the algorithm tries to put c *in the class containing* e, $class(e, P') = \{e, h, i\}$. *Adding* c *to* $\{e, h, i\}$ *implies to also add* d *and* b *into it because of CCE* (c, d) *and* (d, b) *(*d *and* b *are in the smallest subset around* e *according to the* $(++, -)$ *partition value). Unfortunately, the resulting partition* $\{\{a\}, \{b, c, d, e, h, i\}, \{f, g\}, \{j\}\}$ *also contains a FUE,* (c, i).

The algorithm tries to put e *in the class containing* c, $class(c, P') = \{a, b, c, d\}$. *The resulting partition* $\{\{a, b, c, d, e\}, \{h, i\}, \{f, g\}, \{j\}\}$ *does not contain FUEs nor CUEs. This partition is the result of the intuitive algorithm and is represented on Figure 6.6.*

However, we notice that vertex i *is not with its source* f*, and there is no obligation to separate it from its source (the source notion is explained in Section 6.1.1).*

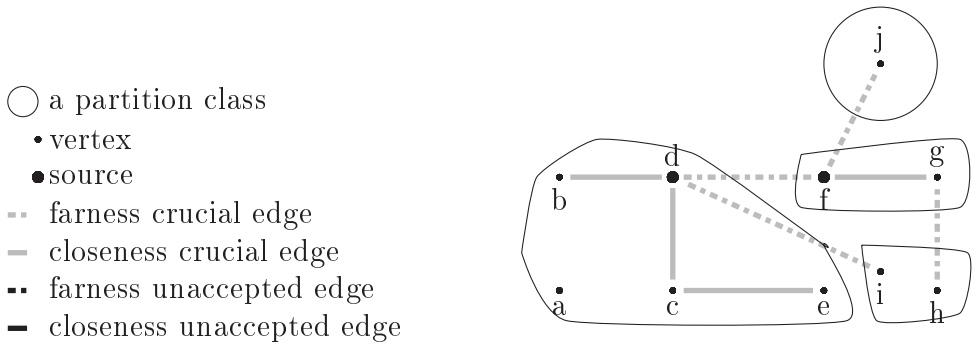

Figure 6.6: The partition P', modification of partition P on $\mathbb{G}_{\mathbb{C}}$ with respect to $(++, -)$ after the merge step of the intuitive repair algorithm (Algorithm 6)

Complexity and transformation number. Let $\mathbb{G}_{\mathbb{C}}$ be a graph, v, a best partition value on $\mathbb{G}_{\mathbb{C}}$, m, be the number of crucial edges, fue, the number of farness unaccepted edges according to v, cue, the number of closeness unaccepted edges according to v, and n the vertexes number.

The ***division step*** of the intuitive repair algorithm is exactly the same as the division step of the naive repair algorithm (Algorithm 5), so it does fue atomic transformations (divisions here) with $\mathcal{O}(m * n)$ complexity in the worst case with m edges and n vertexes, according to Section 6.2.

The ***merge step*** of the intuitive repair algorithm does one transformation per CUE in the worst case. These transformations could be a merge, a replacement or a newplacement. A replacement is worth 2 atomic transformations and a newplacement 3 atomic transformations, so the algorithm does 3 atomic transformations per CUE in the worst case: $3 * cue$ atomic transformations.

The complexity in the worst case is $\mathcal{O}(m*(m+n))$ with m edges and n vertexes. In fact, each CUE is checked (m checks in the worst case), and for each check, one to three "is there a new FUE ?" tests are done (lines 9, 15 and 19). These tests have $\mathcal{O}(m)$ complexity in the worst case because they require to check edges (m), and for each of them, to check the classes of their vertexes (4). Other operations are done, like transformations or identifications of a class. These operations require to check vertexes (n), so, for each tested CUE, there is a complexity of $\mathcal{O}(m+n)$ in the worst case.

The intuitive repair algorithm ***finally*** has $3*m$ atomic transformations (because an unaccepted edge is a farness or a closeness unaccepted edge, but not both) and a complexity of $\mathcal{O}(m*(m+n))$ in the worst case with m edges and n vertexes.

The intuitive repair algorithm repairs the partition, but does not take care of sources (notion explained in Section 6.1.1) and does not try to let vertexes in the same class as their sources as much as possible. The source repair algorithm adds some modifications to the intuitive repair algorithm in the aim of also taking into account sources.

6.4 Source repair algorithm

The source repair algorithm (Algorithm 10) functions as the intuitive repair algorithm (Algorithm 6 Section 6.3) but takes care of sources' positions. As explained in Section 6.1.1, sources are particular vertexes. They are considered as the most significant vertexes of the initial partition Pi. We choose to take care of sources by, for their class, not removing a vertex initially in it, if it can be avoided.

The algorithm stands on the Property 3.

The source repair algorithm (Algorithm 6) uses the source positions when a class has to be divided or merged. The source positions affect transformations of classes. There are three affected operations: division of a class, replacement and newplacement. Let us explain how those operations are affected.

Let Pi, be the initial partition on the criteria graph $\mathbb{G}_\mathbb{C}$,v, a best value on $\mathbb{G}_\mathbb{C}$, $P_{ref} = Pref(v, \mathbb{G}_\mathbb{C})$, and P, a partition on $\mathbb{G}_\mathbb{C}$ and the copy P' of P.

Division step: (lines 2 to 4 from Algorithm 10). The division step of the sources repair algorithm (Algorithm 10) changes the class division of Algorithm 6 a bit with the notion of source position.

Let (x, y) be a farness crucial edge of P' according to v value such that x, y are in the same class c of P'. In the aim of dividing c without separating a vertex $k \in c$ of its source denoted $s = source(k, Pi)$ (please see Section 6.1.1 for details about sources), we have to check if x or y have to be in the same class as s according to classes of P_{ref} (which is equivalent to s being in the smallest subset around x nor y). If none of them requires to be with s, the division will be the same as in previous algorithms 5 and 6. In the other case, the vertex that does not require to be with its own source will be removed from the class (instead of the other requiring to be in the same class as s), as in previous algorithms. The details of a division relative to the source position are in Algorithm 7.

Algorithm 7 sourceDivision

Require: (x, y), an unaccepted farness edge for the partition Pi on the criteria graph $\mathbb{G}_{\mathbb{C}} = (V, E)$ with respect to the partition value $v = v(P_ref)$;
Pi, the initial partition on $\mathbb{G}_{\mathbb{C}}$;
P', a partition on $\mathbb{G}_{\mathbb{C}}$;
v, a possible partition value on $\mathbb{G}_{\mathbb{C}}$;
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}) = v$.

Ensure: a partition resulting from the division of the class containing x and y, relative to source position.

1: Class $result = around(x, P', P_{ref})$;
2: **if** $source(x, Pi) \in result$ **then**
3: $result = around(y, P', P_{ref})$;
4: **end if**
5: **return** $division(result, P')$;

Example 53 (Source division)**.** *Let us take the unaccepted edges of the graph $\mathbb{G}_{\mathbb{C}}$ with respect to partition value v for the initial partition $P = \{\{a, b, c, d\}, \{e, f, g, h, i, j\}\}$ described in Example 49 and represented on Figure 6.3.*

The division step of the sources repair algorithm (Algorithm 10) changes the class division of Algorithm 6 a bit with the notion of source position. The modified partition before the division step is $P' = P$. There are two FUEs: (j, f) and (g, h). Let us divide the class containing j and f, $c = \{e, f, g, h, i, j\}$ because of (j, f). The first vertex is j, and is not linked to any other vertex, so j is put in a new class as for the naive algorithm and P' becomes $\{\{a, b, c, d\}, \{j\}, \{e, g, h, i, f\}\}$ (as in the naive repair algorithm). P' is represented on Figure a) 6.5.

Let us divide the class containing g and h, $c = \{e, g, h, i, f\}$ because of (g, h). Contrary to the naive repair algorithm, we will not put g and f (linked by the CCE (f, g)) in a distinct class because f is the source. So, h is put in a new class and the resulting new P' partition is: $\{\{a, b, c, d\}, \{j\}, \{h\}, \{e, g, i, f\}\}$, represented on Figure 6.7.

There are no more FUEs in P' at the end of the division step.

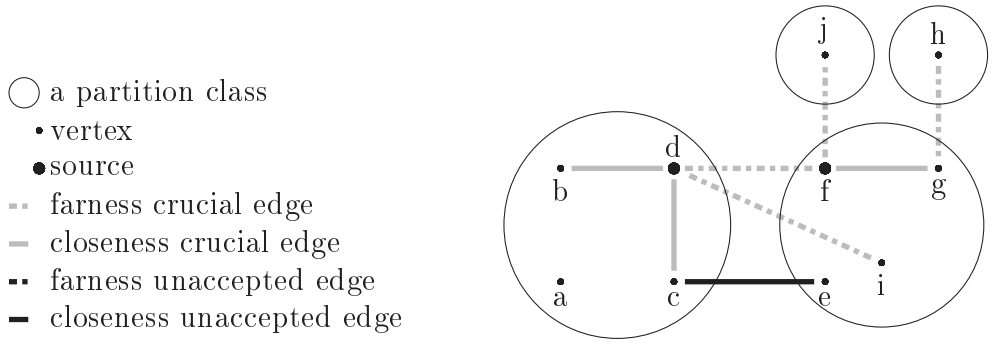

Figure 6.7: The partition P', modification of partition P on $\mathbb{G}_\mathbb{C}$ with respect to $(++,-)$ after the division step of the source repair algorithm (Algorithm 10)

Merge step: (lines 5 to 28 of Algorithm 10) Two transformations are modified with respect to the intuitive repair algorithm (Algorithm 6): replacement and newplacement. Merge is unchanged.

- Replacement (lines 14 to 20) is changed because the smallest subset around a vertex x of a class is changed: if x has to be in the same class as its source $source(x, Pi)$ for any partition with v as partition value (s is in $around(x, P', P_ref)$), then this smallest subset is the entire class of x ($class(x, P')$), else, this is the smallest common subset to $class(x, Pi)$ and $class(x, P_{ref})$. We abusively denote this notion $classSource(x, P', v)$ because P_{ref} and Pi are fixed (please see Algorithm 8).

- Newplacement (lines 21 to 25) is changed if one of the concerned vertexes x, y has to be with its respective source (if only one, because of the classSource notion, like in replacement). If both of them have to be with their respective source, they first are put in a new class as in the intuitive repair algorithm (with $class(x, P') \cap class(x, P_{ref})$ and $class(y, P') \cap class(y, P_{ref})$), and each subset $class(z, P') \cap class(z, P_{ref})$ such that z was in $class(x, P')$ or $class(y, P')$ is added to the new class if this does not add any FUE in the resulting partition. Algorithm 9 details this modification.

Example 54 (Algorithm classSource)**.** *Let P be the initial partition such that $P = \{\{a, b, c, d\}, \{e, f, g, h, i, j\}\}$ the initial partition on $\mathbb{G}_\mathbb{C}$ of the previous Examples. Its unaccepted edges according to the partition value $v = (++,-)$ are represented on Figure 6.3. The class source of g according to v in P is $classSource(g, P) = \{e, f, g, h, i, j\}\}$ because g has to be in the same class as f with respect to v and f is a source, but $classSource(e, P) = \{e\}\}$.*

Example 55 (The merge step of source repair algorithm)**.** *Let us take the unaccepted edges of the graph $\mathbb{G}_\mathbb{C}$ with respect to partition value v for the initial partition $P =$*

Algorithm 8 classSource

Require: x, a vertex of the criteria graph $\mathbb{G}_{\mathbb{C}} = (V, E)$;
Pi, the initial partition on $\mathbb{G}_{\mathbb{C}}$;
P', a partition on $\mathbb{G}_{\mathbb{C}}$,
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$;
Ensure: a subclass of $class(x, P')$ relative to $source(x, Pi)$ position.
1: Class $result = around(x, P', P_{ref})$;
2: **if** $source(x, Pi) \in result$ **then**
3: $result = class(x, P')$;
4: **end if**
5: **return** $result$;

Algorithm 9 newPlacementSource

Require: x, y two vertexes of the criteria graph $\mathbb{G}_{\mathbb{C}}$;
Pi, the initial partition on $\mathbb{G}_{\mathbb{C}}$;
P', a partition on $\mathbb{G}_{\mathbb{C}}$;
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$;
Ensure: a partition with a newplacement of classes $class(x, P')$ and $class(y, P')$ relative to $source(x, Pi)$ and $source(y, P)$ positions, such that there is no new FUE in it.
1: Partition $P_{result} = newplacement(around(x, P', P_{ref}) \cup around(y, P', P_{ref}), P')$;
2: **for all** $z \in class(x, P') \cup class(y, P') | z \notin class(x, P_{result})$ **do**
3: Class $c_z = around(z, P_{result}, P_{ref})$;
4: Partition $P_{test} = replacement(c_z, class(x, P_{result}), P_{result})$;
5: **if** $\nexists (q, w) \in farUnacceptedEdge(v, P_{test}) | (q, w) \notin farUnacceptedEdge(v, P_{result})$ **then**
6: $P_{result} = P_{test}$;
7: **end if**
8: **end for**
9: **return** P_{result};

$\{\{a, b, c, d\}, \{e, f, g, h, i, j\}\}$ *described in Example 49 and represented on Figure 6.3. We took the* P' *partition that is the modified partition after the division step of the source repair algorithm (Example 53):* $P' = \{\{a, b, c, d\}, \{j\}, \{h\}, \{e, g, i, f\}\}$.

The single CUE is (e, c). *As with the intuitive repair algorithm (Example 52), merging classes of* e ($class(e, P') = \{e, g, i, f\}$) *and* c ($class(c, P') = \{a, b, c, d\}$) *is not an option because the FCE* (i, d) *would be unaccepted in the resulting partition. However,* e *can be replaced in* c*'s class. The resulting partition is* $P' = \{\{a, b, c, d, e\}, \{j\}, \{h\}, \{g, i, f\}\}$ *and is represented on Figure 6.8.*

In this Example, the merge step of the source repair algorithm gives a distinct result from the merge step of the intuitive repair algorithm only because the partitions

after the division steps of the algorithms are distinct from each other.

However, j, h and e are in distinct classes in the resulting partition, despite the fact that they are in the same class of the initial partition P and there is no information that forbids them to be in a same class.

Complexity and transformation number. For any partition transformation, the source repair algorithm takes a special care of sources. However, it divides classes a bit too much, as shown in Examples 53 and 55. We propose a way to fix this point in the next advanced repair algorithm.

The source repair algorithm has the same complexity in the worst case as the intuitive repair algorithm ($\mathcal{O}(m * (n + m))$, with m edges and n vertexes).

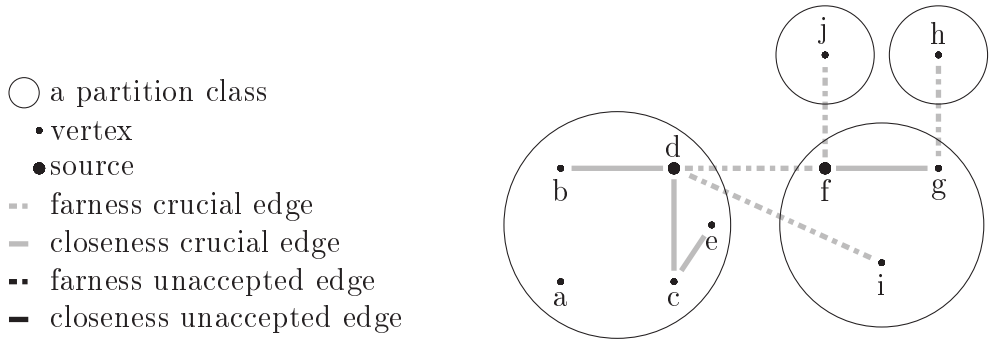

Figure 6.8: The partition P', modification of partition P on $\mathbb{G}_\mathbb{C}$ with respect to $(++, -)$ after the merge step of the source repair algorithm (Algorithm 10)

6.5 Advanced repair algorithm

The advanced repair algorithm compensates the fact that the source repair algorithm divides classes a bit too much (by division, replacement or newplacement transformations), or not always at the most appropriate cut. It is for this reason that the advanced repair algorithm adds an undo-division step to the division and the merge steps of the source repair algorithm (Section 6.4).

In the reminder of this thesis, we will use the notation $classes(c, P)$ to refer to the set of the classes of the partition P that contains at least a vertex included in the vertex set (or class) c.

Undo-divisions step: (lines 2 to 4 of Algorithm 11). Let Pi be the initial partition on the criteria graph $\mathbb{G}_\mathbb{C}$, and v, a partition value on $\mathbb{G}_\mathbb{C}$. P' is the modified partition, resulting from Algorithm 10: P' has no unaccepted edges according to v, but it is possible that some classes have been divided too much. The undo-divisions step checks for each pair of classes c, c' of P' if they have vertexes from a same class

Algorithm 10 sourceRepairAlgorithm

Require: $\mathbb{G}_{\mathbb{C}} = (V, E)$, a criteria graph;
v, a possible partition value for $\mathbb{G}_{\mathbb{C}}$;
Pi, the initial partition on $\mathbb{G}_{\mathbb{C}}$;
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$;
Ensure: a partition P' that has v as partition value (or a better one)

1: Partition $P' = copy(Pi)$;
2: **while** $\exists (x, y) \in farUnacceptedEdge(v, P')$ **do**
3: $P' = sourceDivision((x, y), Pi, P', P_{ref})$;
4: **end while**
5: **while** $\exists (x, y) \in closeUnacceptedEdge(v, P')$ **do**
6: Class $c_x = class(x, P')$;
7: Class $c_y = class(y, P')$;
8: Partition $P_{test} = merge(c_x, c_y, P')$;
9: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | (z, w) \notin farUnacceptedEdge(v, P')$ **then**
10: $P' = P_{test}$;
11: **else**
12: Class $c'_x = classSource(x, Pi, P', P_{ref})$;
13: Class $c'_y = classSource(y, Pi, P', P_{ref})$;
14: $P_{test} = replacement(c'_x, c_y, P')$;
15: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | (z, w) \notin farUnacceptedEdge(v, P')$ **then**
16: $P' = P_{test}$;
17: **else**
18: $P_{test} = replacement(c'_y, c_x, P')$;
19: **if** $\nexists (z, w) \in farUnacceptedEdge(v, P_{test}) | \quad (z, w) \notin farUnacceptedEdge(v, P')$ **then**
20: $P' = P_{test}$;
21: **else if** $source(x, Pi) \in c_x$ and $source(y, Pi) \in c_y$ **then**
22: $P' = newplacementSource(x, y, Pi, P', P_{ref})$
23: **else**
24: $P' = newplacement(c'_x \cup c'_y, P')$;
25: **end if**
26: **end if**
27: **end if**
28: **end while**
29: **return** P';

of the initial partition (ligne 2) and no FUE between them (undo-divisions does not add FUE(s) to the resulting partition). If it is the case, P' becomes $merge(c, c', P')$ and so on until such classes do not exist any more.

This step enables us to compensate cases where the algorithm over divides by merging over-divided classes, as we will see in the next Example.

Algorithm 11 advancedRepairAlgorithm

Require: $\mathbb{G}_{\mathbb{C}} = (V, E)$, a criteria graph;
v, a possible partition value for $\mathbb{G}_{\mathbb{C}}$;
Pi, the initial partition on $\mathbb{G}_{\mathbb{C}}$;¨
P_{ref}, the reference partition on $\mathbb{G}_{\mathbb{C}}$ such that $v(P_{ref}, \mathbb{G}_{\mathbb{C}}) = v$.
Ensure: a partition P' that has v as partition value (or a better one)
1: Partition $P' = sourceRepairAlgorithm(\mathbb{G}_{\mathbb{C}}, v, P)$;
2: **while** $\exists c, c'$ distinct classes of $P' | classes(c, Pi) \cap classes(c', Pi) \neq \{\}$ and $farUnacceptedEdge(v, merge(c, c', P')) = \{\}$ **do**
3: $P' = merge(c, c', P')$;
4: **end while**
5: **return** P';

Example 56 (Advanced repair algorithm)**.** *Let us take the unaccepted edges of the graph $\mathbb{G}_{\mathbb{C}}$ with respect to partition value v for the initial partition $P = \{\{a, b, c, d\}, \{e, f, g, h, i, j\}\}$ described in Example 49 and represented on Figure 6.3. If the resulting partition P' at the end of the division step of the source repair algorithm was the resulting partition $P' = \{\{a, b, c, d\}, \{e, j, h\}, \{g, i, f\}\}$, P' would become $P' = \{\{a, b, c, d, e, j, h\}, \{g, i, f\}\}$ at the end of the merge step of the source repair algorithm because of CUE (d, i). However, P' is the partition $\{\{a, b, c, d, e\}, \{j\}, \{h\}, \{g, i, f\}\}$ represented on Figure 6.8, as explained in Example 55 because j, h and e were extracted from their original class $\{e, f, g, h, i, j\}$ separately.*

The undo-division step undoes that by merging:

- *$\{j\}$ and $\{h\}$ (there is no FUE between them and $class(j, Pi) = class(h, Pi) = \{e, f, g, h, i, j\}$. P' becomes $P' = merge(\{j\}, \{h\}, P')$ and is represented on Figure 6.9. $P' = \{\{a, b, c, d, e\}, \{j, h\}, \{g, i, f\}\}$*

- *$\{a, b, c, d, e\}$ and $\{j, h\}$ (there is no FUE between them and $class(e, Pi) = class(j, Pi) = \{e, f, g, h, i, j\}$. P' becomes $P' = merge(\{a, b, c, d, e\}, \{j, h\}, P')$ and is represented on Figure 6.10. $P' = \{\{a, b, c, d, e, j, h\}, \{g, i, f\}\}$.*

There are no more P' classes to merge because there are farness crucial edges between the last two classes, like (i, d). The two merges committed by the undo-division step enable us to compensate two of the three divisions committed by the division and merge steps. In fact, the merge step uses transformations like replacement and newplacement which imply one or two divisions.

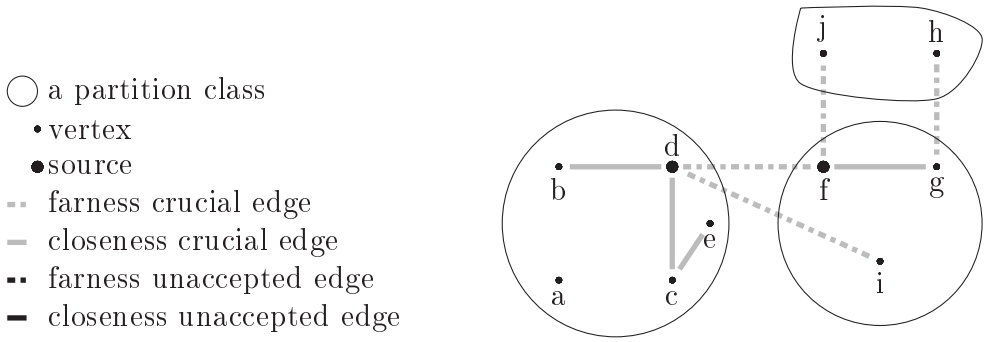

Figure 6.9: The partition P', modification of partition P on $\mathbb{G}_{\mathbb{C}}$ with respect to $(++,-)$ after the first merge in the undo division step of the advanced repair algorithm (Algorithm 10)

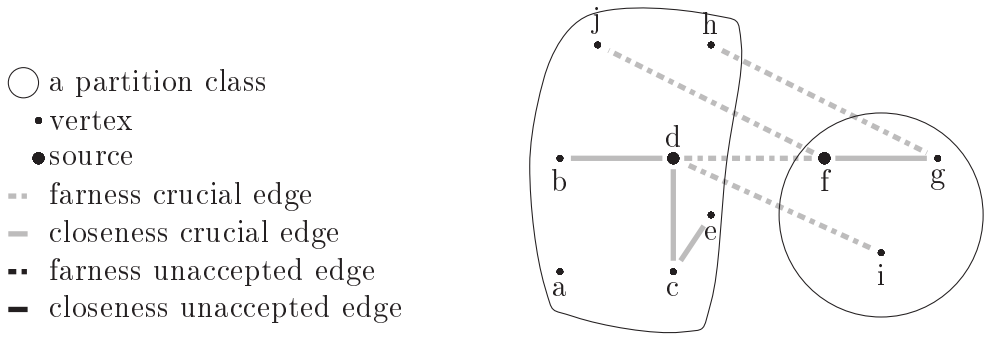

Figure 6.10: The partition P', modification of partition P on $\mathbb{G}_{\mathbb{C}}$ with respect to $(++,-)$ after the undo-division step of the advanced repair algorithm (Algorithm 10)

Complexity and transformation number. The advanced repair algorithm adds 1 atomic transformation per unaccepted edge in the worst case (to the 3 atomic transformations per unaccepted edge of the source repair algorithm), that avoids an atomic transformation (division) previously done by the source repair algorithm. So, the resulting partition has been less divided than the resulting partition of the source repair algorithm, as shown in Example 56.

Let n be the number of vertexes and m be the number of edges. The undo-division step checks each pair of vertexes ($n*(n-1)/2$ checks in the worst case). For each pair of vertexes, if they are in distinct classes, it checks if there is a farness edge between these classes (m edges to check in the worst case) and may merge these classes (n operations in the worst case). As a consequence, the complexity of the undo-division step is $\mathcal{O}(n^2(m+n))$ in the worst case. The advanced algorithm also uses the source repair algorithm ($\mathcal{O}(m*(n+m))$ complexity in the worst case),

so its total complexity in the worst case is $\mathcal{O}((n^2 + m) * (m + n))$.

We presented in this Chapter four algorithms for modifying a partition in order to improve its partition value. They can be used to repair erroneous links in a bibliographic knowledge base such as Sudoc. Each of them has been theoretically evaluated in its own Section. Let us evaluate in the next Chapter the partitioning semantics used for detecting erroneous links and proposed in Chapter 5.

Chapter 7

Experiments

In this Chapter we will evaluate the results of global and local partitioning semantics. The repair algorithms have been theoretically evaluated in Chapter 6. Section 7.3 will show that the proposed algorithms are efficient to find the best partition values on real Sudoc subsets[1]. Then, we will study how data characteristics influence the results of these algorithms in Section 7.2 before checking that partitioning semantics are able to detect erroneous links by comparing initial and human partition[2] values on a real Sudoc sample in Section 7.3. We will finish by presenting a future interface that will be used by experts in order to see erroneous links in Section 7.4.

7.1 Quantitative experiments

In this Section, we present experiments that show that algorithms used to find the best partition values according to global and local semantics (please see Chapter 5 for details about these semantics) are efficient on real Sudoc subsets.

We have experimented the algorithms used to find all the best partition values according to global and local semantics (described in Chapter 5) on 1796 Sudoc subset related to 1796 random appellations. Those appellations have been randomly chosen[3]. For each appellation, we select the Sudoc subset related to it.

We recall that the complexity in the worst case for finding the best partition values according to global semantics (Algorithm 3 page 73) is $\mathcal{O}((k+1)^c * m \log n)$

[1] We recall that a Sudoc subset related to an appellation A is the set of contextual entities that represents a link to an authority notice with an appellation close to A. This notion has been detailed in Section 4.1.5 page 47.

[2] We recall that the initial partition is the partition that reflects links in Sudoc before repairing links and that the human partition reflects good links according to a human expert. Those notions have been defined in Section 3.2.3 page 37.

[3] Each Sudoc authority notice is stored in its own file. Those files have been classified by using a key that depends on each character in the file, with a lot of variability. 1796 keys have been selected. We took the first appellation of each authority notice contained by a file such that its key was selected. Those appellations are the 1795 randomly chosen appellations.

with n vertexes, m edges, c criteria, and k the maximal number of closeness values of a criterion in the considered criterion set and that the complexity in the worst case for finding the best partition values according to local semantics (Algorithm 4 page 81) is $\mathcal{O}(n * (k+1)^c * m \log n)$. For both algorithms, $(k+1)^c$ corresponds to the number of reference partitions to evaluate in the worst case.

Each selected Sudoc subset contains 1 to 1238 contextual entities. We did not represent on Figures the 7 Sudoc subsets that contained more than 653 authority notices because they are too few to be significant. We evaluated the execution time for a criterion set $\mathbb{C}$ of six criteria ($\mathbb{C} = \{appellation, language, date, domain, title, otherContributors\}$) described in Section 4.2.2 page 50. Each of these criteria has 0 to 6 closeness values (there are 84 possible closeness value sets for $\mathbb{C}$, so 84 reference partitions to evaluate in the worst case to ensure to have every best partition value with respect to global semantics).

We used a Intel(R) Core(TM) i7-2600 CPU 3.40 GHz PC with 4GB of RAM running Windows 7 64 Bit with a Java 1.6 implementation. The execution times are shown on Figures 7.2 and 7.4.

Before executing the algorithms used to find all the best partition values according to global semantics (Algorithm 3 page 73) and local semantics (Algorithm 4 page 81), let us observe the distribution of links in Sudoc.

Distribution of contributor links in Sudoc. In order to look at the link distribution in Sudoc, Alain Gutierrez enumerated the number of bibliographic notices linked per authority notice per role (as "author"). The number of links per authority notices is represented on Figure 7.1. The scale is not linear because links are distributed in a very unequal fashion. As shown on Figure, the role "author" is widely predominant over other roles. For "author" links, we observe that:

- 1520285 authority notices are linked at least once;
- 972 authority notices are linked at least 250 times;
- 113 authority notices are linked at least 1001 times.

Links are distributed in a very unequal fashion among authority notices. Few authority notices are very linked (113), and the large majority are linked to few bibliographic notices (1425473 authority notices are linked to 10 bibliographic notices at most by an "author" link). So, it is reasonable to consider that only a little part of Sudoc subsets related to an appellation contains more than 350 contextual entities.

Global semantics (Algorithm 3 page 73). As shown on Figure 7.2, execution times for Algorithm 3 (which finds the best partition values according to global semantics and several criteria) are fast:

- less than 6 seconds for up to 653 authority notices;

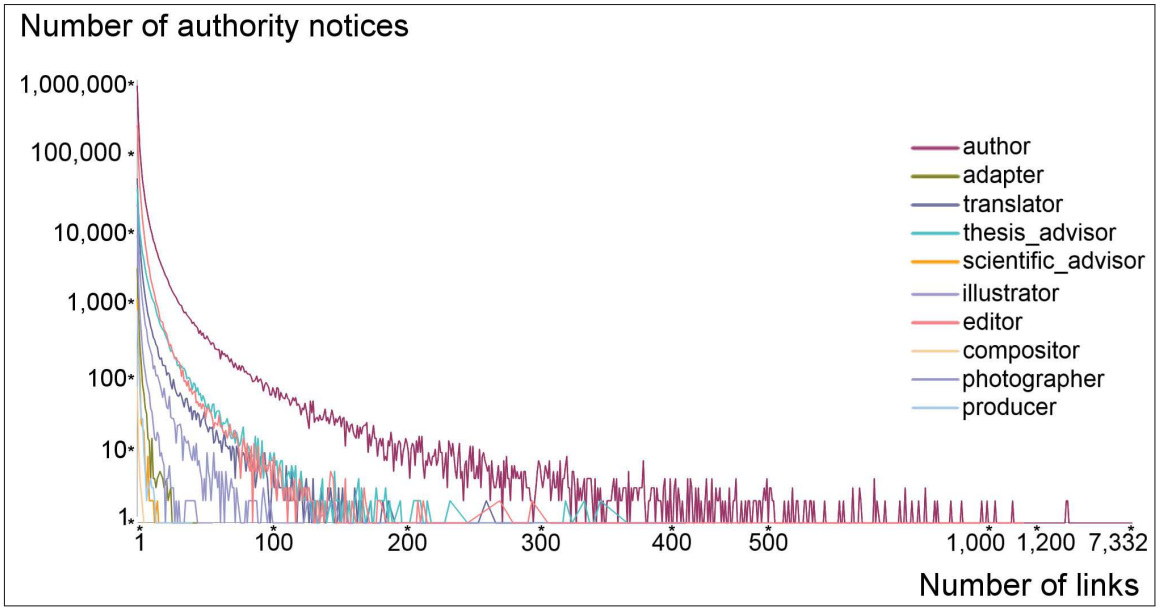

Figure 7.1: Links distribution between Sudoc authority notices

- less than 2 seconds for up to 444 authority notices;
- less than 1 second for up to 231 authority notices.

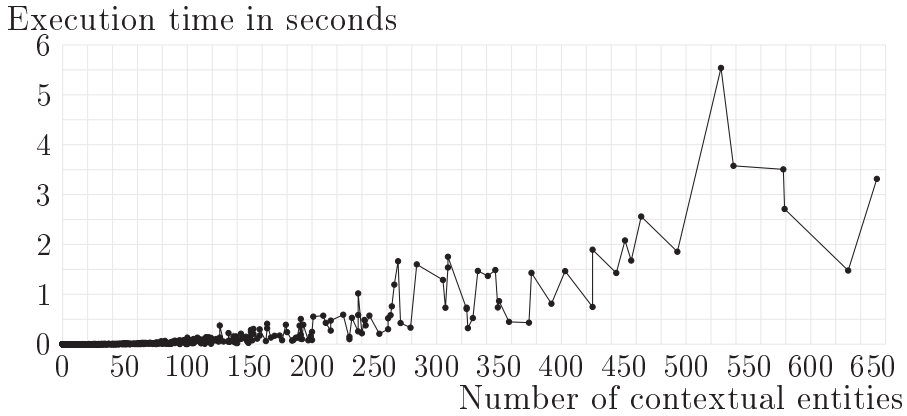

Figure 7.2: Execution time for global semantics algorithm

Those results are good: it is reasonable to consider that there are only a few Sudoc subsets contain more than 350 contextual entities. Indeed, we saw on Figure 7.1 that there are only 972 authority notices that are linked to 250 or more bibliographic notices as an "author" in the entire Sudoc.

The repartition of Sudoc subsets of the studied sample according to their size (number of contextual entities) is shown on Figure 7.3. This Figure confirms that Sudoc subsets have often a small size: 1,194 in 1796 Sudoc subsets contain 10 contextual entities at the most, and only 26 in 1796 Sudoc subsets contain over 350 contextual entities.

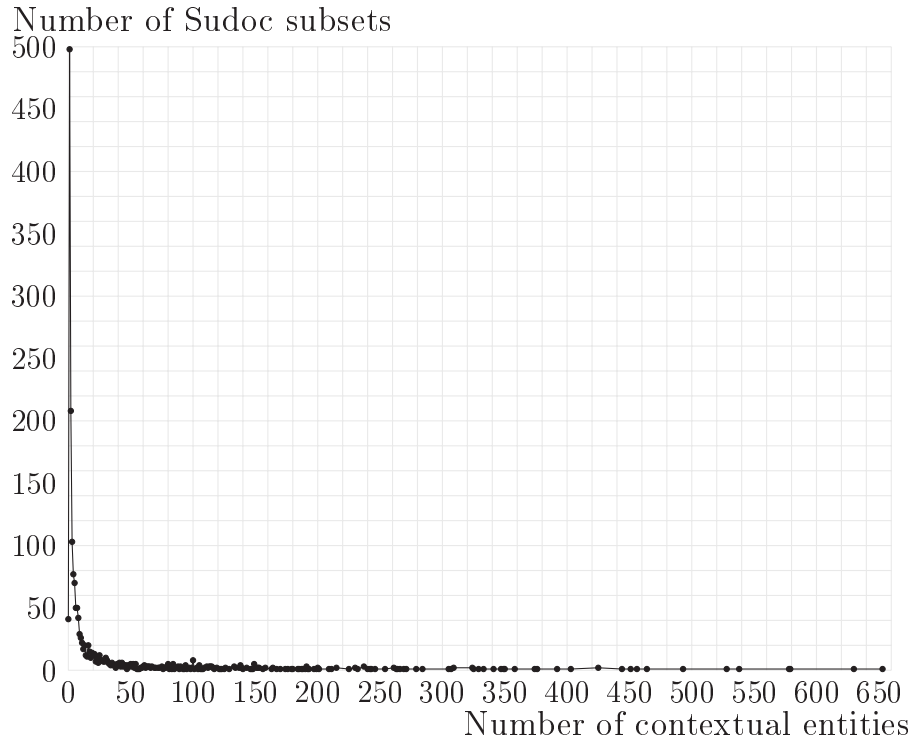

Figure 7.3: Sudoc subsets size

Figure 7.2 shows spikes. This is due to the fact that Algorithm 3 has to calculate and evaluate a number of reference partitions which goes from 1 to 84 for the used criterion set. This depends on incoherences[4] in the Sudoc subset evaluated, as we will see in Section 7.2.2.

Local semantics (Algorithm 4 page 81). The execution time to find all the best partitions according to local semantics algorithm is shown on Figure 7.4 and looks depicted. The local semantics improves global semantics by adding the notion of locality, but the execution time to find all the best partition values is longer than the one for the global semantics algorithm. Indeed, the complexity is also greater for the algorithm used to find best partition values according to local

[4]We recall that an incoherence in an object set according to a criterion set $\mathbb{C}$ is when two objects must be in distinct classes according to one criterion of $\mathbb{C}$ and in the same class according to some criteria of $\mathbb{C}$. Incoherences are explained in Section 5.3.1 page 76.

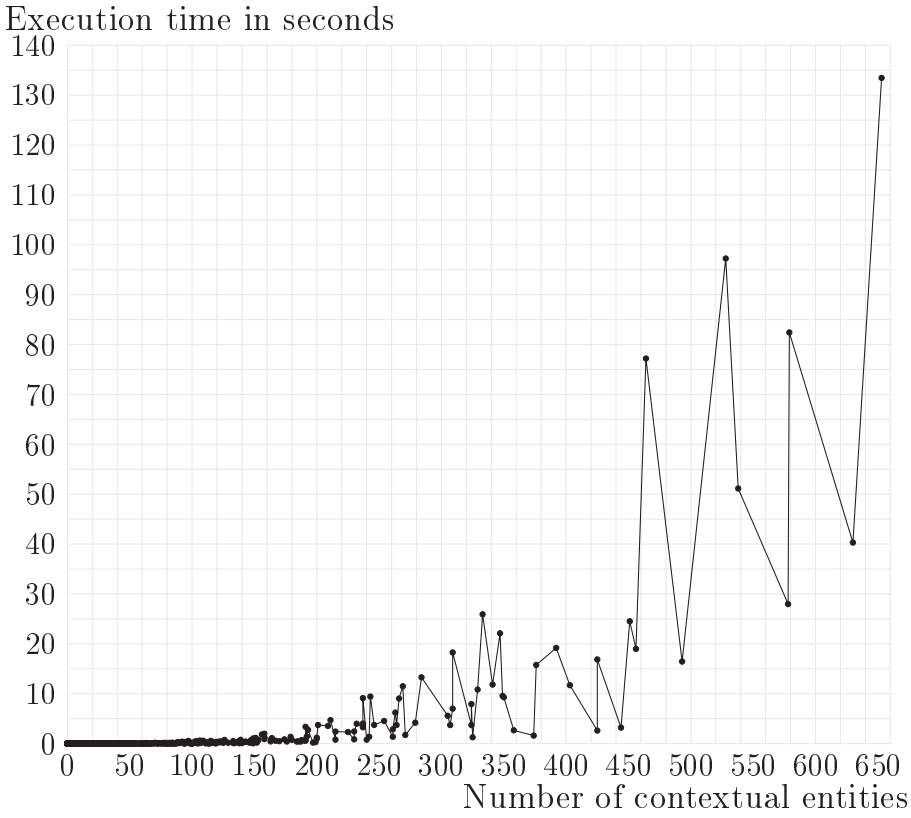

Figure 7.4: Execution time for local semantics algorithm

semantics ($\mathcal{O}(n * (k+1)^c * m \log n)$ with n vertexes, m edges, c criteria, and k the maximal number of closeness values of a criterion in the considered criterion set) than according to global semantics ($\mathcal{O}((k+1)^c * m \log n)$). The results are acceptable (less than 6 seconds) for Sudoc subsets that contain less than 232 contextual entities but are not acceptable for larger Sudoc subsets. Indeed, the algorithm needs 22 seconds at worst for Sudoc subsets containing up to 349 contextual entities or 97 seconds for Sudoc subsets containing up to 349 contextual entities.

In this Section we saw that algorithms used to find the best partition values according to global or local semantics are efficient enough for being used on most of real Sudoc subsets. Let us see how the choice of criteria can improve their efficiency.

7.2 Quantitative experiments: data influence

In this Section, we study how input characteristics (data ambiguity, accuracy of the criteria, number of incoherences) influence the partitioning semantics' output (human partition value, number of incoherences and algorithm execution time).

Considered input characteristics are:

- **data ambiguity:** We say that **data** is **ambiguous** if there are distinct real-world entities that have the same name and contributed to some very similar real-world documents.

- **the accuracy of the criteria:** The criteria can lack precision, and consider as similar distinct contextual entities (or consider as far close contextual entities).

- **the number of incoherences:** please see Section 5.3.1 page 76 for details about incoherences. As we will see, the number of incoherences is a consequence of data ambiguity and the accuracy of the criteria. It influences result characteristics.

Considered output characteristics are:

- **the value of the human partition:** We recall that the human partition is the partition of contextual entities such that the links are validated by a human expert. This notion is explained in Definition 2 page 37).

- **the number of incoherences** (please see Section 5.3.1 page 76 for details about incoherences).

- **the execution time** of algorithms *globalAlgorithm* (Algorithm 3 page 73) and *localAlgorithm* (Algorithm 4 page 81).

In the reminder of this Section, we define each data characteristics and explore their consequences on the characteristics of the results. We consider a contextual entity set $\mathbb{O}$ and a criterion set $\mathbb{C}$. Used criteria are detailed in Section 4.2.2 page 50.

7.2.1 Data ambiguity and the accuracy of the criteria

Let us define data ambiguity and the accuracy of the criteria in order to explain how they influence number of incoherences.

Definition 34 (Data ambiguity)**.** *The **data ambiguity** is a measure of how many pairs of $\mathbb{O}$ objects are in an object set $\mathbb{O}$ such that they seem very close (respectively far) but are not in real life.*

Using the above introduced notation, for a set of contextual entities $\mathbb{O}$, data ambiguity corresponds to how many contextual entities Nc_i, Nc_j of $\mathbb{O}$ exist such that $C(Nc_i)$ and $C(Nc_j)$ seem to represent the same (respectively distinct) real-world person but do not.

The more ambiguous data is, the closer (or farther) pairs of objects seem to each other but are not. **Increasing data ambiguity could increase the number of incoherences**.

Definition 35 (Accuracy of a criterion)**.** *The* ***accuracy of a criterion*** *c is the ratio of the pairs of objects well compared with respect to real-world (the criterion gives an appropriate closeness, farness, never or always comparison value) to the pairs of objects compared with a different than neutral comparison value. For a given criterion set $\mathbb{C}$, the* ***accuracy of the criteria*** *is the average accuracy of $\mathbb{C}$ criteria.*

Let us see how the accuracy of the criteria influences results.

The less accurate criteria are, the more it happens that pairs of objects compared, according to a criterion, with a closeness or *always* comparison value (respectively farness or *never* comparison value) are actually far (respectively close) from each other. This could increase the number of incoherences.

The less accurate criteria are, the more ambiguous data seems, and, the more ambiguous data is, the more difficult it is to make accurate criteria in order to compare the objects.

We saw that decreasing the accuracy of criteria or increasing data ambiguity increases the number of incoherences. Let us see what is influenced by the number of incoherences in the following Section.

7.2.2 Number of incoherences

In an object set $\mathbb{O}$ according to a criterion set $\mathbb{C}$, it is hard to directly measure the number of incoherences[5]. Indeed, to measure the number of incoherences implies to calculate every path between every pair of objects linked by a farness edge.

We will consider in this Section that decreasing the accuracy of the criteria increases the number of incoherences (explained in Section 7.2.1), and will use it to observe the effects of increasing the number of incoherences on several characteristics of the results (i.e. human partition value, execution time of *localAlgorithm* and *globalAlgorithm*, and number of incoherent parts).

Human partition having a best partition value

In order to detect erroneous links, the initial partition[6] is supposed to be a best partition if and only if it corresponds to the human partition. That implies that the human partition must have a best partition value, according to Work Hypothesis 7 (page 20).

The human partition is supposed to have a best partition value. If the initial partition corresponds to the human partition and does not have a best value, that could be because of the number of incoherences as shown in the next Example.

[5] The Incoherence notion is detailed in Section 5.3.1 page 76.

[6] As explained in Definition 1 page 37, the initial partition is the partition such that the contextual entities are in the same class if and only if they represent a link pointing at the same authority notice. This partition represents the state of the Sudoc before repairing links.

Example 57 (Human partition value and number of incoherences)**.** *Let $\mathbb{O}_{nm}$ be the contextual entities Sudoc subset related to "*NICOLAS, *Maurice". Let $\mathbb{C}$ be the criterion set $\mathbb{C}$ such that $\mathbb{C} = \{date, title, appellation, otherContributors, language, domain\}$. According to global semantics, there are 2 best partition values (the first is good according to all criteria but language, the second is good for all criteria, except title and otherContributors) and the human partition has a best partition value.*

Let us add incoherences by decreasing the accuracy of criteria. Let $20\%randomDate$ be a new criterion such that, for two contextual entities with a date attribute in both of them, there is a 20% chance to give a random comparison value (neutral, a closeness or a farness value). In other cases, it is the same comparison value as the original date criterion (neutral or a farness value). We calculated all best partition values and evaluated the human partition on $\mathbb{O}_{nm}$ for $\mathbb{C}' = (\mathbb{C} \cup \{20\%randomDate\} - \{date\})$ and found 4 best partition values. The human partition did not have a best value according to $\mathbb{C}'$.

We can thus see that increasing the number of incoherences could make the human partition not have a best partition value, but it could also increase the number of best partitions, as we will see in the next Section.

The number of best partition values

Let us consider an object set $\mathbb{O}$ with no data ambiguity, and a perfectly accurate criterion set $\mathbb{C}$ such that $\mathbb{C}$ criteria do not consider as close (respectively far) some objects that should not be in the same class (respectively in distinct classes). There is a single best partition value that is the best one according to all criteria. If we add an incoherence, by making a criterion $c \in \mathbb{C}$ consider this pair of objects as far (respectively close) despite other criteria's advice, there are two best partition values:

- the one that is good according all criteria but not c and
- the one that is good according to c but not to other criteria.

Adding more incoherences potentially adds more best partition values, but decreases these values.

On the contrary, when there are a lot of incoherences, (especially those involving *always* or *never* values), that reduces the valid (and possible) partition values because some of the most intense values of some criteria are incoherent with some of the most intense (or *always* and *never*) values of other criteria. In extreme cases, the best partition values cannot better satisfy a criterion over others but can only satisfy closeness values (corresponds to the partition with a single class) or farness values (corresponds to the partition with a class per object), if there are no *always* or *never* values involved. If there are *always* and *never* values involved in the same incoherence, we have no more best partition values because there are no more valid partitions.

A lot of incoherences can decrease the number of best partition values. Let us show that in the following Example.

Example 58 (Number of best values and the accuracy of the criteria)**.** *We reuse the "*NICOLAS*, Maurice" object set $\mathbb{O}_{nm}$, $\mathbb{C}$ and $\mathbb{C}'$ (respectively $\mathbb{C}''$) criterion sets of Example 57. The human partition of $\mathbb{O}_{nm}$ (denoted Ph_{mn}) has 4 best partition values with respect to the criterion set $\mathbb{C}$ and global semantics. date has been replaced by 20%randomDate (respectively 100%randomDate, which has a random comparison value in all cases) and measured 25 times (because 20%randomDate and 100%randomDate are random criteria) the number of the best partition values with respect to the criterion set $\mathbb{C}'$ (respectively $\mathbb{C}''$) and global semantics. Results are:*

- *with 20%randomDate criterion: 1 to 4 best partition values, 3.0 on average;*
- *with 100%randomDate criterion: 1 to 2 best partition values, 1.04 on average.*

So, artificially increasing the number of incoherences (by decreasing criterion accuracy) a little has increased the number of best partition values for $\mathbb{O}_{nm}$ object set, but increasing it by a lot reduces that number to nearly 1, which is the smallest possible number of the best partition values with our criterion set (because a partition is always valid according to this criterion set).

Consequently, the number of incoherences influences the number of the best partition values. Let us now see how the number of incoherences influences execution time for algorithms that find all the best partition values for both local and global semantics. We start by looking at the execution time for *globalAlgorithm*.

The execution time of *globalAlgorithm* (Algorithm 3 page 73)

The **execution time for an algorithm** is the time, in milliseconds, taken by the algorithm at hand to return a result.

Increasing the number of incoherences increases the execution time of *globalAlgorithm* by requiring more reference partitions[7] to calculate and evaluate in order to find out all the best partition values (because less reference partitions have an optimal value due to incoherences, and this implies that their descendants have to be calculated and evaluated).

Example 59 (Number of incoherences and execution time for the *globalAlgorithm* algorithm)**.** *In this Example, we took the object set related to appellation "*LEROUX*, Alain" and measure the execution time[8] according to 8 criteria: appellation,*

[7] We recall that reference partitions are the partition required to be evaluated in order to find best values. However, it is not necessary to evaluate all of them in all cases, as explained in Chapter 5. Please see Definition 22 page 72 for details.

[8] We used a Intel(R) Core(TM) i7-2600 CPU 3.40 GHz PC with 4GB of RAM running Windows 7 64 Bit with a Java 1.6 implementation.

title, language, otherContributors, role, thesis, thesisAdvisor and randomDate′. The randomDate′ is a criterion as 20%*randomDate criterion detailed in Example 57 but the percentage of random comparison values varies according to "percent of randomDate′" of Figure 7.5. For each tested percentage (from* 0 *to* 100% *by steps of* 5*) of random comparison:*

- *we generated 25 times the randomDate′ criterion and measured every time the execution time (represented by a gray dot);*
- *we calculated the average mark of those 25 execution time measures and represented it by a black dot on Figure.*

We can see on Figure 7.5 that the ***execution time increases with the percentage of random comparisons*** *of randomDate′ criterion from 1 milliseconds on average for 0% of random comparisons (when randomDate′ corresponds to date criterion) to 9 milliseconds on average for 100% of random comparisons. Also, for a given percentage of random comparisons, the execution time only slightly fluctuates. There is a single tiny spike at 5% of random comparisons, which could be explained by the fact that execution times fluctuate more for 5% of random comparisons than for 10% to 20% of random comparisons.*

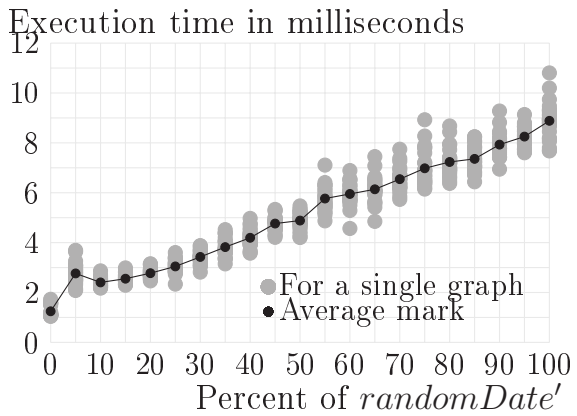

Figure 7.5: Execution time to find all best partition values according to global semantics and 8 criteria including *randomDate′*

Therefore, we can conclude that **increasing the number of incoherences increases execution time for** *globalAlgorithm* in a nearly monotonous manner. Before seeing how the number of incoherences influences *localAlgorithm*'s execution time, we first need to see **how the number of incoherences influences the number of incoherent parts**.

The number of incoherent parts

Incoherent parts, as explained in Section 5.3.1 page 76, are independent parts that contain at least an incoherence. Therefore, adding incoherences can, at the same time:

- **increase the number of incoherent parts** (by adding a farness or *never* value to an independent part not incoherent yet, it will add an incoherence in it and make it a new incoherent part);
- **reduce the number of incoherent parts** (by adding a closeness or *always* comparison value between two objects from two incoherent parts, which will merge them and decrease the total number of incoherent parts by 1).

Example 60 (Incoherences and number of incoherent parts)**.** *In this Example, we consider the object set corresponding to appellation "*Leroux*, Alain" and the criterion set of Example 59. For each tested percentage randomDate′:*

- *we took the 25 randomDate′ criteria generated in Example 59 and measured the number of incoherent parts, which is represented on Figure 7.6 by a grey dot;*
- *we calculated the average mark of those 25 incoherent parts measures and represented it by a black dot on Figure.*

For small percentages of random comparison (5% to 15%), the number of incoherent parts fluctuates a lot, in particular for 5% and when the average number of incoherent parts is more than 1. For no random comparisons or 20% and more random comparisons, there is a single incoherent part all the time.

That Example shows that **decreasing slightly the number of incoherences increases the number of incoherent parts** by adding incoherences in independent parts, but **increasing a lot the number of incoherences does not increase the number of incoherent parts** because most of the independent parts are merged in a single big incoherent part.

The execution time of *localAlgorithm*

For the same reason that the number of incoherences increases the execution time of *globalAlgorithm*, the **number of incoherences also increases the execution time of** *localAlgorithm*.

However, **the execution time of** *localAlgorithm* **also depends on incoherent parts** because it executes *globalAlgorithm* for each incoherent part, as shown in Section 5.3 page 76. So, the execution time of *localAlgorithm* also increases with the number of incoherent parts, which depends on the number of incoherences, as shown in Section 7.2.2.

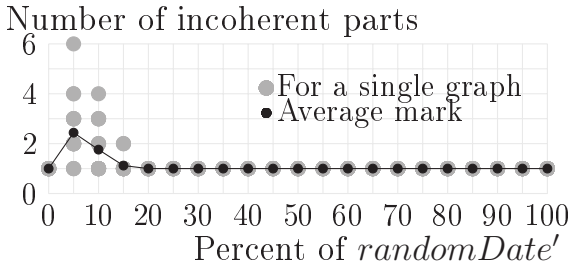

Figure 7.6: The number of incoherent parts according to 8 criteria including *randomDate′*

Example 61 (About the execution time of *localAlgorithm*)**.** *In this Example, we took the object set corresponding to appellation "*Leroux*, Alain" and the criterion set of Example 59. For each tested percentage randomDate′:*

- *we took the 25 randomDate′ criteria generated in Example 59 and each time measured the execution time of localAlgorithm, which is represented by a grey dot on Figure 7.7;*
- *we calculated the average mark of those 25 execution time measures and represented it by a black dot on Figure 7.7.*

The execution times monotonously increase except for 5 to 15% of random comparison values, from 2 milliseconds on average for 0% to 14 milliseconds on average for 100%.

The spike on 5% to 15% exactly corresponds to the increased number of incoherent parts shown on Figure 59. We also notice that execution time fluctuates a lot on these percentages, but fluctuates slightly for other percentages.

For a given number of incoherent parts, increasing the number of incoherences increases the execution time for *localAlgorithm*.

Conclusion. As a conclusion to this Section, the number of incoherences increases with data ambiguity and decreases with the accuracy of the criteria. The number of incoherences greatly influences executions time of the algorithms, the number of best partitions and if the human partition has a best partition value.

Let us qualitatively evaluate the global and local partitioning semantics.

7.3 Qualitative experiments

In this Section, we will be interested in the qualitative results of the local and global partitioning semantics presented in Chapter 5.

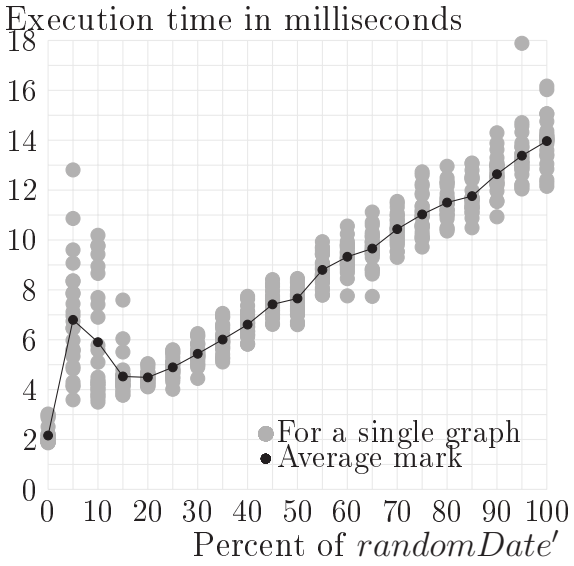

Figure 7.7: Execution time to find all best partition values according to local semantics and 8 criteria including $randomDate'$

ABES experts selected 537 bibliographic notices linked to 82 authority notices which correspond to 7 appellations. This sample was given in two versions:

- the initial version (the links between bibliographic notices and authority notices are as they were when the sample was selected);
- the human version (the links between bibliographic notices and authority notices were manually repaired by the experts).

The sample is divided into 7 Sudoc subsets related to an appellation (1 per appellation). Each Sudoc subset contains the contextual entities representing a link between an authority that could represent a person named by the appellation and a bibliographic notice linked to such an authority notice. The human partition on a Sudoc subset is deduced from the human version, and the initial partition is deduced from the initial version of the Sudoc sample chosen by the experts.

Let us define and evaluate the **precision** (Definition 36), the **recall** (Definition 37) and the F-measure (Definition 38) of both semantics in order to detect erroneous links in this sample. Intuitively, the precision is useful to determine if a test finds as positive only objects that should be. The recall is useful to determine if a test finds as positive all the objects that should be. The F-measure is an equilibrium between precision and recall.

Definition 36 (Precision). *Let $\mathbb{O}$ be an object set such that objects can be positive for a concept C. The set of positive objects is denoted $\mathbb{O}_p$ and the set of objects found by a test T as positive for the concept C is denoted $\mathbb{O}_f$. The precision of the test T on $\mathbb{O}$ is: $precision(T, \mathbb{O}) = \frac{|\mathbb{O}_p \cap \mathbb{O}_f|}{|\mathbb{O}_f|}$.*

Definition 37 (Recall). *Let $\mathbb{O}$ be an object set such that objects can be positive for a concept C. The set of positive objects is denoted $\mathbb{O}_p$ and the set of objects found by a test T as positive for the concept C is denoted $\mathbb{O}_f$. The recall of the test T on $\mathbb{O}$ is: : $recall(T, \mathbb{O}) = \frac{|\mathbb{O}_p \cap \mathbb{O}_f|}{|\mathbb{O}_p|}$.*

Definition 38 (F-measure). *Let $\mathbb{O}$ be an object set such that objects can be positive for a concept C, T, a test for the concept C, precision, the precision of T on $\mathbb{O}$ and recall, the recall of T on $\mathbb{O}$.*

The F-measure of the test T on $\mathbb{O}$ is: $F{-}measure(T, \mathbb{O}) = \frac{2 * precision(T,\mathbb{O}) * recall(T,\mathbb{O})}{(precision(T,\mathbb{O}) + recall(T,\mathbb{O}))}$.

7.3.1 Evaluation of the initial partitions according to both semantics

Let us check if the global and local semantics allow us to detect erroneous links in the initial partition.

Table 7.1 shows for each Sudoc subset related to the chosen appellation (in "appellation"):

1. "Size" shows the number of contextual entities that represent a link between a bibliographic notice and an authority notice that has a close appellation to the chosen appellation,

2. "Is Pi best?" shows whether the initial partition Pi issued from the initial version is valid or has a best value according to global semantics and with respect to the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$. Those criteria are detailed in Section 4.2.2 page 50.

3. "Is Ph better than Pi? " is true if and only if the human partition Ph has a better value than Pi according to global semantics and with respect to the criterion set $\mathbb{C}_9$.

Table 7.2 shows the same info as Table 7.1 but according to local semantics instead of global semantics.

Local semantics has the same results as global semantics on this sample.

We first evaluated the partition values (Pi and Ph for each Sudoc subset for both semantics according to the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$. The initial partition (Pi) has a partition value that is worse than the human partition (Ph) for each

Table 7.1: Evaluation of Pi according to global semantics

Appellation	Size	Is Pi best?	Is Ph better than Pi?
"Bernard, Alain"	165	not valid	yes
"Dubois, Olivier"	27	no	yes
"Leroux, Alain"	59	not valid	yes
"Roy, Michel"	52	not valid	yes
"Nicolas, Maurice"	20	no	yes
"Simon, Alain"	63	no	yes
"Simon, Daniel"	151	not valid	yes

Table 7.2: Evaluation of Pi according to local semantics

Appellation	Size	Is Pi best?	Is Ph better than Pi?
"Bernard, Alain"	165	not valid	yes
"Dubois, Olivier"	27	no	yes
"Leroux, Alain"	59	not valid	yes
"Roy, Michel"	52	not valid	yes
"Nicolas, Maurice"	20	no	yes
"Simon, Alain"	63	no	yes
"Simon, Daniel"	151	not valid	yes

Sudoc subset tested, and is not even valid[9] 4 times out of 7 for both semantics. The human partition (Ph) is always valid. When a partition is not valid, it is because two contextual entities have the same title (and so, correspond to a same work[10]) but are in distinct classes (details about the *title* criterion are available in Section 4.2.2 page 50).

Example 62 (Partition not valid)**.** *Let us consider the Sudoc subset related to "*ROY*, Michel". This Sudoc subset contains at least two authority notices that are in distinct classes of the initial partition Pi and forbid Pi to be valid:*

- *the contextual entity representing the link between the bibliographic notice having the ppn "005643309" and the title "Les techniques psycho-corporelles : de la relaxation au stretching" and the authority notice having the ppn "134443985" representing the contributor "*ROY*, Michel";*
- *the contextual entity representing the link between the bibliographic notice having the ppn "004065468" and the title "Les techniques psycho-corporelles : de la relaxation au stretching" and the authority notice having the ppn "030093422" representing the contributor "*ROY*, Michel".*

For global semantics or local semantics, the initial partition Pi (deduced from the initial version of Sudoc that has wrong links) is never a best partition, so the precision and the recall are 100% for detecting erroneous links in initial partitions for both semantics.

Let us evaluate the human partitions on the Sudoc subsets issued from the human version for each semantics.

7.3.2 Evaluation of the human partitions according to the global semantics

We evaluated the human partitions on each Sudoc subset (deduced from the human version of the Sudoc sample) according to the global semantics and the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$. Those human partitions have also been evaluated with respect to the criterion set $\mathbb{C}_7$, which corresponds to $\mathbb{C}_9$ without *domain* and *language* criteria. Results are summarized in Table 7.3:

1. "Size" is the number of contextual entities that represent a link between a bibliographic notice and an authority notice that has a close appellation to the chosen appellation,

[9] The validity of a partition is defined Definition 11 page 61. A partition that is not valid cannot be a best partition and is not even considered as a solution for the entity resolution problem.

[10] As explained in Section 4.1.3 page 43, Sudoc bibliographic notices represents manifestations, an edition of a document. Manifestations could correspond to a same work, which implies that their contributors with a close appellation are the same real-world person.

Table 7.3: Global semantics: human partition evaluation with respect to $\mathbb{C}_9$

Appellation	Size	Number of Na	Ph best?
"Bernard, Alain"	165	27	no
"Dubois, Olivier"	27	8	no
"Leroux, Alain"	59	6	no
"Roy, Michel"	52	9	no
"Nicolas, Maurice"	20	3	yes
"Simon, Alain"	63	13	no
"Simon, Daniel"	151	16	no

Table 7.4: Global semantics: human partition evaluation with respect to $\mathbb{C}_7$

Appellation	Number of Nc	Number of Na	Ph partition value
"Bernard, Alain"	165	27	yes
"Dubois, Olivier"	27	8	no
"Leroux, Alain"	59	6	yes
"Roy, Michel"	52	9	no
"Nicolas, Maurice"	20	3	yes
"Simon, Alain"	63	13	no
"Simon, Daniel"	151	16	yes

2. "Number of Na" is the number of authority notices according to human partitions (corresponding to the number of classes in the human partitions),

3. "Ph best?" shows whether the human partition Ph issued from the human version has a best value according to global semantics with respect to the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$.

Table 7.4 shows the same info as Table 7.3 but according to the criterion set $\mathbb{C}_7 = \mathbb{C}_9 - \{language, domain\}$).

We can see in Table 7.3 that there is only 1/7 of the Sudoc subset on which the human partition Ph (deduced from the human version) has a best partition value according to global semantics and the criterion set $\mathbb{C}_9$. Removing *domain* and *language* from the criterion set improves such Sudoc subsets to 4/7 as we can see in Table 7.4. So, we have a recall of 4/7 or 57% and a precision of 100% for detecting the partition value as a best partition. Let us detail the exceptions.

The human partition Ph on Sudoc subset related to "Roy, Michel" does not have a best value with respect to $\mathbb{C}_7$ because two contextual entities have a common contributor who is not "Roy, Michel", but are not in the same class. This other contributor is represented by the authority notice having the ppn "031684769". This

is wrong according to the criterion *otherContributor*[11]. Those contextual entities represent the following links:

- the link between the bibliographic notice having the ppn "165760214" and the title "*La médaille : technique de fabrication*" and the authority notice having the ppn "168583275" representing "ROY, Michel";
- the link between the bibliographic notice having the ppn "129161942" and the title "*Maille, medaglia, médaille*" and the authority notice having the ppn "168580969" representing "ROY, Michel".

The ABES expert has confirmed that there is an erroneous link.

The human partition *Ph* on Sudoc subset related to "SIMON, Alain" does not have a best value with respect to $\mathbb{C}_7$ because the "SIMON, Alain" represented by the authority notice having the ppn "030480388" is supposed to have supervised a thesis [12] in "1988", one year before submitting his own thesis[13] in "1989". This is wrong according to criterion *thesisAdvisor*.

According to the ABES expert, those contextual entities may represent distinct persons. In this case, they recommend to link the corresponding bibliographic notices to distinct authority notices.

The human partition *Ph* on Sudoc subset related to "DUBOIS, Olivier" does not have a best value with respect to $\mathbb{C}_7$ because the "DUBOIS, Olivier" represented by the authority notice having the ppn "09293580X" is supposed to have supervised two theses [14] in "1989", three years before submitting his own thesis [15] in "1992". This is wrong according to criterion *thesisAdvisor*.

According to the ABES expert, those contextual entities must represent distinct persons, and there is an erroneous link.

---

[11] Criteria are detailed in Section 4.2.2 page 50.

[12] represented by the bibliographic notice having the ppn "043118186" and the title "*PLACE DES VASODILATATEURS DANS LA PHARMACOLOGIE ANTIHYPERTENSIVE*"

[13] represented by the bibliographic notice having the ppn "04322928X" and the title "*LES ABCES DU FOIE A PYOGENES : A PROPOS D'UNE SERIE DE DIX PATIENTS*"

[14] represented by the bibliographic notice having the ppn "043179924" with "*CESARIENNE AVANT TRAVAIL ET HYPERTENSION ARTERIELLE DE LA GROSSESSE : RETENTISSEMENT NEONATAL A PROPOS DE 474 CAS D HTA ASSOCIEE A LA GROSSESSE, ETUDE PARTICULIERE DES CAS AYANT JUSTIFIE UNE CESARIENNE*" as title, and the bibliographic notice having the ppn "043254381" with "*LE TABAGISME CHEZ LA FEMME ENCEINTE DANS LA REGION NORD/PAS-DE-CALAIS : FREQUENCE, CONSEQUENCES FOETALES ET NEONATALES*" as title

[15] represented by the bibliographic notice having the ppn "040921018" with "*L'AFFIRMATION DE SOI EN SERVICE D HOSPITALISATION PSYCHIATRIQUE : MISE EN PLACE D UN GROUPE THERAPEUTIQUE ET REFLEXION CRITIQUE AU TERME D UN AN DE PRATIQUE*" as title

According to the ABES experts, all human partitions on Sudoc subset that were not evaluated as best partitions by global semantics contain erroneous links. They are not considered human partitions any more. According to the ABES expert, those contextual entities must represent distinct persons, and there is an erroneous link, so the **recall** is 4/4 (100%) and not 4/7 (57%) to identify the human partition as a best partition according to global semantics and the criterion set $\mathbb{C}_7$.

Let us evaluate the recall and precision of the local semantics for identifying the human partition as a best partition.

7.3.3 Evaluation of the human partitions according to the local semantics

In the same way we evaluated the human partition according to global semantics on the Sudoc sample in the previous Section, we have evaluated the human partitions on each Sudoc subset (deduced from the human version of the Sudoc sample) according to the local semantics and the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$. Those human partitions have also been evaluated with respect to the criterion set $\mathbb{C}_7$, which corresponds to $\mathbb{C}_9$ without *domain* and *language* criteria. Results are summarized in Table 7.5:

1. "Size" is the number of contextual entities that represent a link between a bibliographic notice and an authority notice that has a close appellation to the chosen appellation,

2. "Number of $|Na|$" is the number of authority notices according to human partitions (corresponding to the number of classes of the human partitions),

3. "Ph evaluation" shows the number of parts of the considered Sudoc subset for which the human partition has a best value according to local semantics[16] on the total number of parts (with respect to the criterion set $\mathbb{C}_9 = \{title, otherContributors, thesis, thesisAdvisor, date, appellation, language, role, domain\}$).

Table 7.6 shows the same info as Table 7.5 but according to the criterion set $\mathbb{C}_7 = \mathbb{C}_9 - \{language, domain\}$).

We can see that there are often several parts for a Sudoc subset according to the criterion set $\mathbb{C}_9$ in Table 7.5, so there are often incoherent parts according to $\mathbb{C}_9$. We can see in Table 7.6 that there is always a single part for a Sudoc subset according to $\mathbb{C}_7$ there is always a single part. Each of these single parts is the coherent part on all the Sudoc subsets of the sample. So, in this case the results are exactly the same as for global semantics.

[16] We recall that the local semantics divides the criteria graph into coherent parts and incoherent parts. There is always a single coherent part and the criterion graph may have several incoherent parts, or none. Details about local semantics are available in Chapter 5.

Table 7.5: Local semantics: human partition evaluation with respect to $\mathbb{C}_9$

Appellation	Size	Number of Na	Ph evaluation
"Bernard, Alain"	165	27	0/2
"Dubois, Olivier"	27	8	0/1
"Leroux, Alain"	59	6	1/2
"Roy, Michel"	52	9	0/3
"Nicolas, Maurice"	20	3	1/1
"Simon, Alain"	63	13	0/2
"Simon, Daniel"	151	16	0/4

Table 7.6: Local semantics: human partition evaluation with respect to $\mathbb{C}_7$

Appellation	Size	Number of Na	Ph evaluation
"Bernard, Alain"	165	27	1/1
"Dubois, Olivier"	27	8	0/1
"Leroux, Alain"	59	6	1/1
"Roy, Michel"	52	9	0/1
"Nicolas, Maurice"	20	3	1/1
"Simon, Alain"	63	13	0/1
"Simon, Daniel"	151	16	1/1

Table 7.7: Precision, recall and F-measure with respect to $\mathbb{C}_9$

Test	Precision	Recall	F-measure
Detecting erroneous links in Pi	100%	100%	100%
Recognizing the Ph as erroneous links free	100%	25% 40%	

Table 7.8: Precision, recall and F-measure with respect to $\mathbb{C}_7$

Test	Precision	Recall	F-measure
Detecting erroneous links into Pi	100%	100%	100%
Recognizing the Ph as erroneous links free	100%	100%	100%

Furthermore, we can deduce that criteria in $\mathbb{C}_7$ have a good accuracy exept for the *domain* and *language* criteria, which are in $\mathbb{C}_9$ and not in $\mathbb{C}_7$. Indeed, we saw in Section 7.2.2 that a bad accuracy of the criteria may increase the number of incoherent parts.

Results are similar to results according to global semantics with respect to both criterion sets $\mathbb{C}_9$ and $\mathbb{C}_7$ for the considered Sudoc subset. The single exception is for the Sudoc subset related to "LEROUX, Alain" with respect to $\mathbb{C}_9$:

- Ph does not have a best partition value according to global semantics, as seen in Table 7.3.
- Ph has half a best partition value (for one of its two parts) according to local semantics, as seen in Table 7.5, but the Ph partition value remains not a best one.

7.3.4 Conclusion

The precision of both semantics with respect to the criterion sets $\mathbb{C}_7$ and $\mathbb{C}_9$ is very good (100%) to detect erroneous links in initial partitions or to identify the human partition as a best one. The recall is similar on the studied sample and depends on the criteria used (between 25% with respect to $\mathbb{C}_9$ and 100% without *domain* and *language* in the criterion set). Let us summarize the resulting precision, recall and F-measure with respect to $\mathbb{C}_9$ in Table 7.7 and with respect to $\mathbb{C}_7$ in Table 7.8 for the following issues

- detecting erroneous links before human repair ("Detecting erroneous links into Pi") and
- recognizing the human partitions as the best partitions or without erroneous links ("Recognizing the Ph as without erroneous links").

The evaluation did not permit to conclude that one semantics gives better qualitative results than the other. According to Section 7.1, the Algorithm for global semantics shows a shorter execution time than the Algorithm for local semantics in finding all best partition values. However, we have shown in Section 5.3.2 page 80 that local semantics allows us to evaluate partitions over several Sudoc subsets together or separately without changing the results, contrary to global semantics for which results depend on how Sudoc subsets have been selected.

Those semantics allow us to determine if the links in a Sudoc subset have to be revised. It also allows us to easily evaluate the accuracy of criteria by giving the number of incoherent parts or the pairs of objects that should be in the same class (respectively in distinct classes) but that are not according to some specific criteria. So, the expert can look at these objects and decide whether it is the partition that is wrong or the criterion that is not accurate enough. This property helped us to detect remaining erroneous links in a sample corrected by experts.

7.4 User interface proposal

In order to help librarians to see Sudoc erroneous links detected by the algorithms, we proposed and discussed an interface. This interface is in French. Let us present it in this Section.

Figure 7.8 shows the interface statement after the librarian has selected the appellation "HERMÈS, Pierre". This interface allows the librarian to select an appellation (first name and name) and to validate it with the "Ok" button.

A Sudoc subset is shown. Then, the interface shows contextual entities and authority notices corresponding to the Sudoc subset related to the selected appellation and comparison values according to criteria. A note says if there are erroneous links or not. In the case of Figure 7.8, there are erroneous links and the note is red and reads "Liens erronés !". The contextual entities that represent a link to the same authority notice are represented together inside a box that represent the authority notice they are linked to. Appellations of authority notices and titles of contextual entities are always shown, but other attributes are available by moving the cursor on the object.

Between two contextual entities, there are at most two lines to represent edges: a dotted line for farness and *never* labelled edges and a full line for closeness and *always* labelled edges. The color of the line is:

- green if all edges inside are satisfied,
- orange if they are not satisfied but no edge labelled *never* or *always* is represented,
- red if they are not satisfied and at least an edge labelled *never* or *always* is represented.

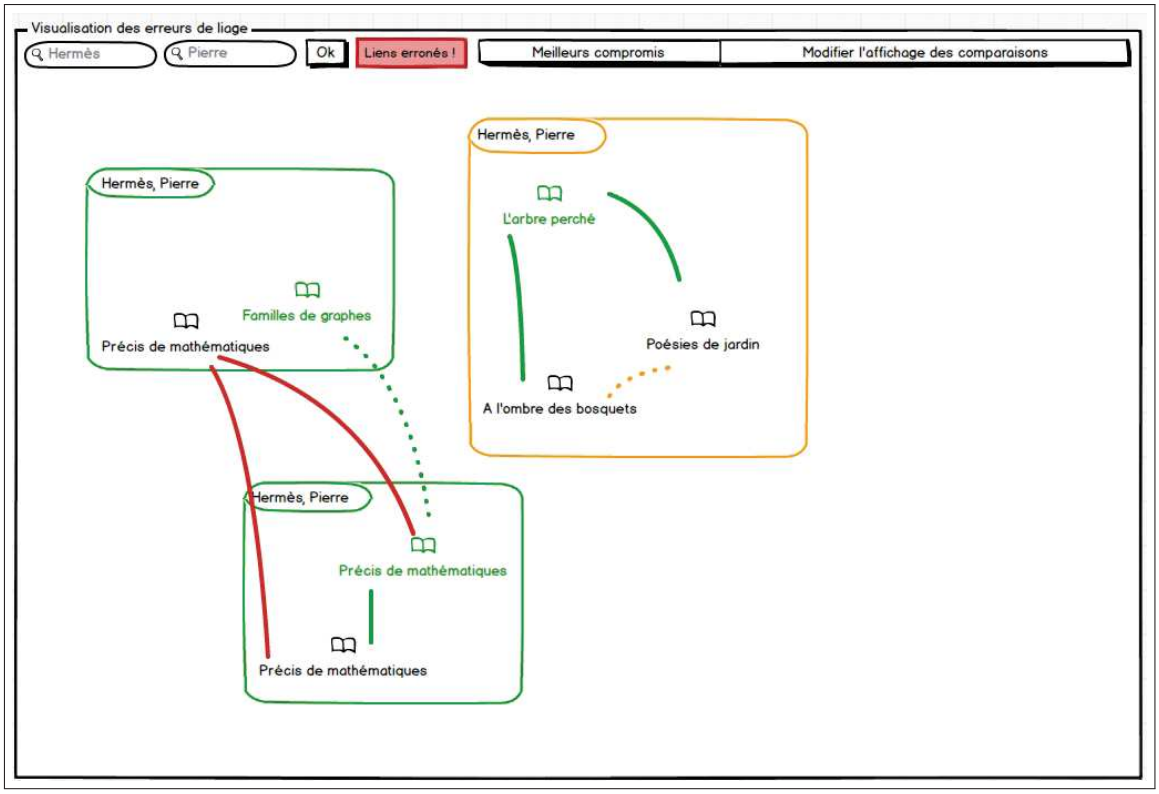

Figure 7.8: Interface for detecting erroneous links with an appellation selected

Moving the cursor on the line shows the labels of each represented edge (value and criterion). The colour of an authority notice box depends on the edges inside it. It is green if all edges inside are satisfied, orange if at least a (farness) edge inside is unsatisfied or red if at least a *never* labelled edge inside is unsatisfied.

At this stage, all edges are represented. Hopefully, two menus allow the librarian to choose which labelled edges (s)he does not want to see.

The "Best compromises" menu (or "`Meilleur compromis`" in French) proposes to show all edges or all crucial edges with respect to a best partition value. All the best partition values are enumerated in common language. Figure 7.9 shows this menu after the librarian has selected to show all crucial edges according to the partition value which does not satisfy the date criterion. We observe that the farness unsatisfied edge between contextual entities titled "*A l'ombre des bosquets*" and "*Poésie de Jardin*" is no more visible on Figure 7.9 but was on Figure 7.8.

The "Modifying the comparisons showing" menu (or "`Modifier l'affichage des comparaisons`" in French) proposes, for each criterion, to check which types of edges have to be shown. The type of an edge corresponds

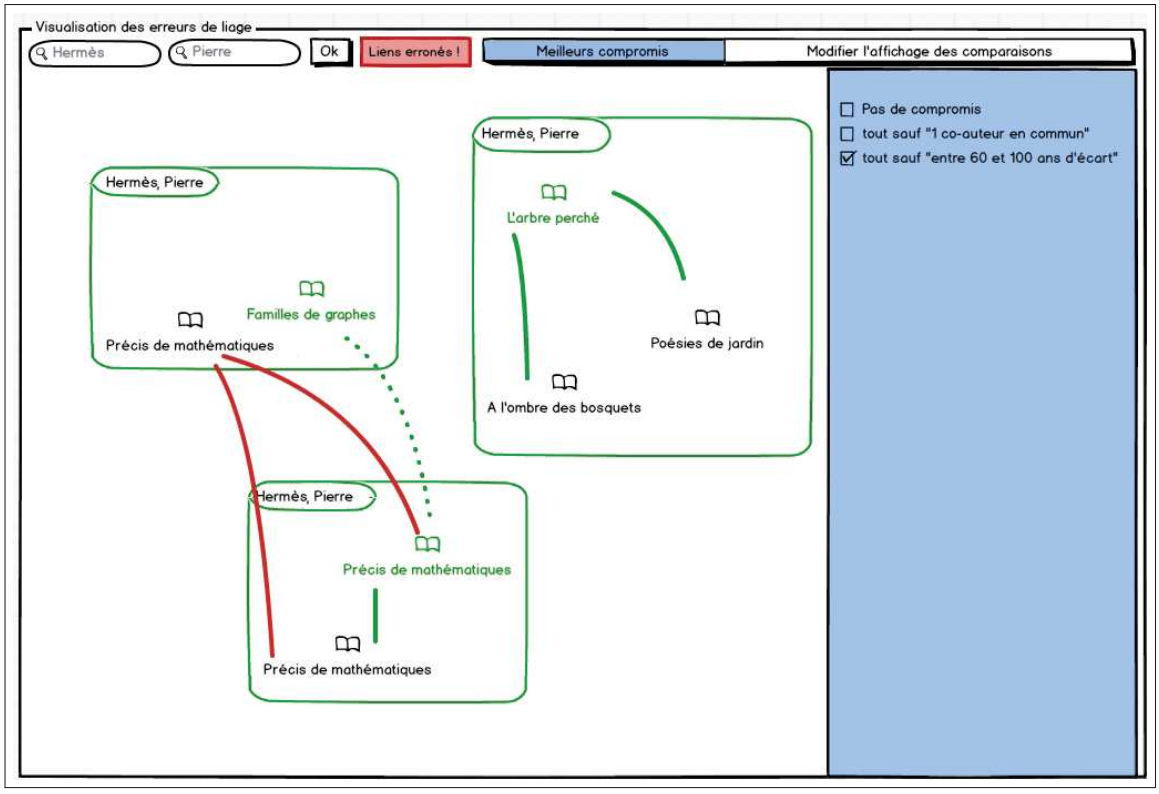

Figure 7.9: Menu "`Best compromises`"

to the intensity of the label. It is expressed in common language. For example, the comparison value *always* of *title* criterion becomes "`same work`" ("`même oeuvre`" in French).

Figure 7.9 shows this menu after the librarian has deselected edges labelled – for *otherContributors* criterion. It corresponds to the interface "`1 or several other contributors in common`" ("`1 ou plusieurs co-auteurs en commun`" in French) unchecked and "`several other contributors in common`" ("`plusieurs co-auteurs en commun`" in French) checked.

We observe that the two closeness satisfied edges between contextual entities titled "*A l'ombre des bosquets*" and "*L'arbre Perché*" and between "*L'arbre Perché*" and "*Poésie de Jardin*" are no more visible on Figure 7.8 but were on Figure 7.9.

Conclusion We showed in this Chapter that algorithms for finding the best partitions according to global and local semantics are efficient on real Sudoc subsets. We show that their results and execution time depend on the number and accuracy of criteria. We also showed that they are relevant for detecting erroneous links in real Sudoc subsets on condition that the considered criteria are relevant. Hopefully these partitioning semantics are also useful to detect criteria with a low accuracy.

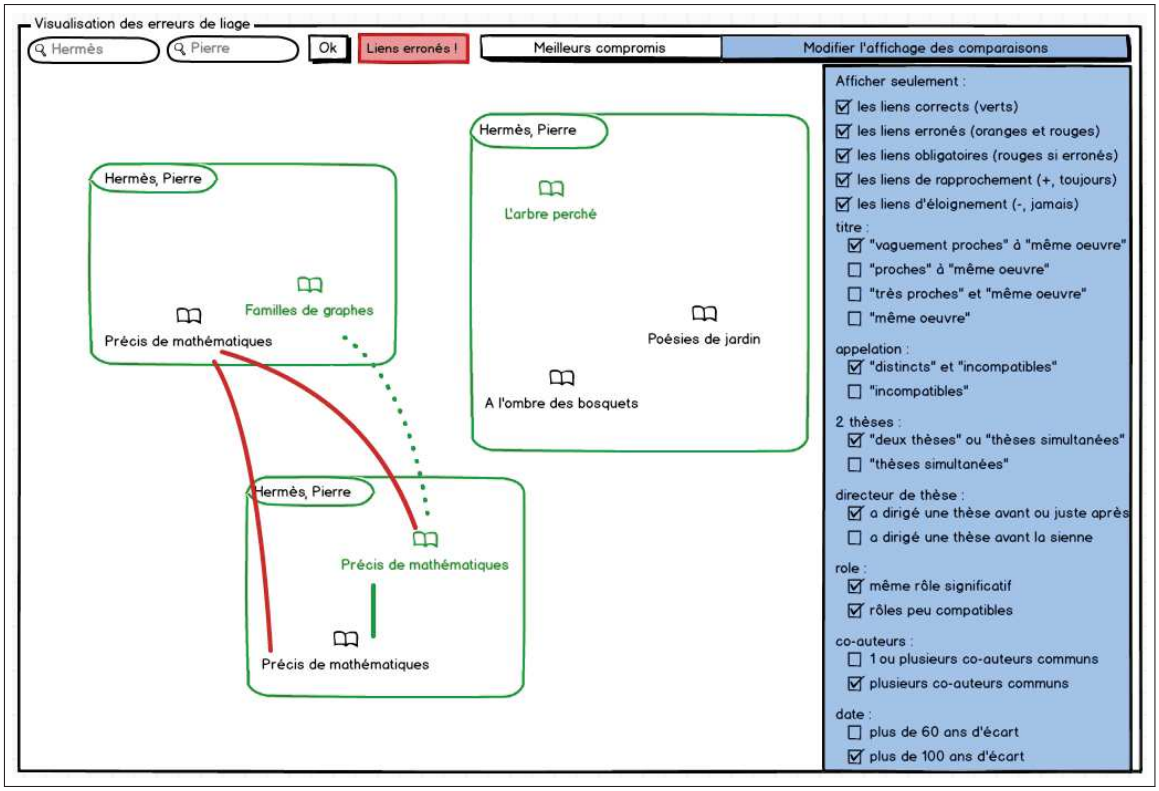

Figure 7.10: Menu "`Modifying the comparisons showing`"

We finally presented a future interface which will be used by librarians to easily access to erroneous links. In time, this interface will be extended to help librarians to repair those erroneous links.

Chapter 8

Conclusion

In this thesis we presented our contribution to the detection and repair of erroneous links. This problem is a link mining problem but, as explained in Chapter 3, our approach uses an entity resolution method in order to solve it. As a consequence, the link mining and entity resolution domains are both presented in Chapter 2. The approach is adapted to bibliographic knowledge bases and applied to Sudoc. Sudoc and its modelization are presented in Chapter 4. The approach is decomposed in two major steps. The detection of erroneous links uses a partitioning semantics based on symbolic criteria. Such partitioning semantics are proposed in Chapter 5. Algorithms for repairing erroneous links are proposed and theoretically validated in Chapter 6. Finally, Chapter 7 presents the qualitative and quantitative experiments on real Sudoc data for the validation of partitioning semantics. It also proposes an interface for human users. This interface is in discussion with experts.

This Chapter concludes the thesis by summarising the achievements of my work in Section 8.1 and presenting future research directions in Section 8.2.

8.1 Research achievements

We presented the research question of this work in Chapter 1: *"How to detect and repair erroneous links in a bibliographic knowledge database using qualitative methods?"*

In order to answer this question, we focused on contextual entities instead of links, authority notices or bibliographic notices (which are already in the bibliographic knowledge base unlike contextual entities). Indeed, contextual entities are objects that represent links (between an authority notice and a bibliographic notice) or a person in the context of a document. This allowed us to detect which links have to point to the same authority notice (or to distinct authority notices) using a partitioning semantics, as explained in Chapter 3.

In order to do that, we proposed two partitioning semantics (local and global semantics) using symbolic criteria. They improve in particular [2] by considering edges labelled *neutral* in the graph to partition. The local semantics adds a notion

of independence between object sets to the global semantics. That allowed us not to change the result when partitioning independent subsets together, like Sudoc subsets related to distinct appellations in our applied case. We have shown in Chapter 7 that those partitioning semantics have good qualitative results to solve the problem of detecting erroneous links, and are useful to detect criteria with a poor accuracy.

However, detecting erroneous links is not enough to improve the link quality in a bibliographic knowledge base. It is for this reason that we proposed several repair algorithms in Chapter 6. We theoretically proved that they correctly repair erroneous links in the same Chapter.

We finally presented an interface that is in discussion with experts and will be used in order to show the erroneous links to librarian users.

8.2 Future work

Let us present several research questions that may improve this work.

Chapter 2 highlights the problem of how adding the notion of canopies to the erroneous link detection method. Indeed, our approach exploits co-contributions links in order to detect erroneous links. So, repairing erroneous links may allow us to detect other erroneous links, and this aspect may be fully exploited by adding a canopy strategy [68] that works under the hypothesis that repairing some links in a canopy (a Sudoc subset related to an appellation in our case) may change the results in neighbour canopies (Sudoc subsets related to the appellation of a co-contributor).

Chapter 5 allows us to detect erroneous links with considering a criterion set of independent criteria. Removing the Work Hypothesis of independence among criteria (Work Hypothesis 6 page 19) allows to us add notions of preferences among criteria.

The interface presented in Chapter 7 may be improved by enable visualization of large Sudoc subsets and repair of erroneous links using the interface.

Let us develop two of these points. The first one is to remove the Work Hypothesis of independence among criteria in Section 8.2.1, and the second is about improving the interface in order to enable visualization of large Sudoc subsets, and will be developed in Section 8.2.2.

8.2.1 Removing the hypothesis of independency among criteria

The partitioning semantics presented in Chapter 5 used a set of symbolic criteria, and those criteria were independent (according to Work Hypothesis 6 page 19). To introduce preferences among criteria in the partitioning semantics may make the results easier to use by letting the semantics choose automatically some criteria over others without requiring the advice of an expert.

In order to do that, we can propose two kinds of preferences:

- to formalize the dependency among comparison values when comparing two objects. Indeed, in our criterion set, the comparison value given by the *thesis* criterion in order to compare two objects depends on the comparison value given by the *title* criterion. Similarly, the comparison value given by any criterion in order to compare two objects depends on the comparison value given by the *appellation* criterion.

- to propose to add the priority relation among criteria as in [14] for comparing two partition values. Let us detail this point.

In order to consider the priority relation among the criteria of a criterion set $\mathbb{C}$, the criteria of $\mathbb{C}$ are separated into n ordered criterion sets such that any criterion of the criterion set $\mathbb{C}_i$ is preferred to any criterion in the criterion set $\mathbb{C}_j$ (denoted $\mathbb{C}_i \gg \mathbb{C}_j$) if and only if $i < j$, and each criterion is in a single criterion set $\mathbb{C}_k$ with $1 \leq k \leq n$. Two valid partitions[1] on the same object set are first compared according to the criterion set $\mathbb{C}_1$. If the values are strictly identical, the value according to $\mathbb{C}_2$ are also considered, and so on until $\mathbb{C}_n$.

In order to find best partitions according to a criterion set and global semantics while considering the priority relation among criteria, two methods can be proposed. The first is to calculate all best partition values like for a set of independent criteria, then to sort partition values while considering the priority relation. The second method is the following one.

Let $\mathbb{C} = \bigcup \mathbb{C}_i$ be a criterion set such that criterion subsets $\mathbb{C}_i$ are independent criterion sets, pairwise disjoint and such that $\mathbb{C}_i \gg \mathbb{C}_j$ if and only if $i < j$. We first find best reference partition[2] values with respect to $\mathbb{C}_1 \in \mathbb{C}$. Then, for each best partition value, we find the corresponding reference partition P_a. Then, we find best partition values with respect to $\mathbb{C}_2$ that have the same value as a best reference partition with respect to $\mathbb{C}_1$ (using the Definition of enemy classes defined in Definition 39), and so forth until the last subset $\mathbb{C}_i \in \mathbb{C}$ (Algorithm 12).

We always take care of *never* and *always* values of all criteria of $\mathbb{C}$, so the calculated best partition values are valid.

Definition 39 (Enemy classes)**.** *Let P be a partition and $\mathbb{C}$ a criterion set on an object set $\mathbb{O}$. Two P classes are enemies for $\mathbb{C}$ if to merge them make the resulting partition to be worse with respect to at least a criterion of $\mathbb{C}$.*

[1] We recall that a valid partition is a partition without unsatisfied edges labelled *never* or *always*, and is formally defined in Definition 11 page 61.

[2] We recall ourselves that best partition values are reference partition values, as explained in Section 5.2.3 page 67, and that the notion of reference partition is defined in Definition 22 page 72.

Algorithm 12 globalAlgorithmWithPriority

Require: an object set $\mathbb{O}$; a criterion set $\mathbb{C} = \bigcup \mathbb{C}_i$ such as the d criterion subsets $\mathbb{C}_i$ possess independent criteria among them, are pairwise disjoint, $\mathbb{C}_k \gg \mathbb{C}_j$ if and only if $k < j$, and with $C \in \mathbb{C}_0$, a criterion that contains all *always* or *never* labelled edges.

Ensure: the set of best partition values on $\mathbb{O}$ according to $\mathbb{C}$ and global semantics with priority.

set of best reference partitions for priority criteria $bestPA = \{P | P$, best reference partition for $\mathbb{C}_0\}$;
set of best reference partitions for criteria that are currently explored $bestPB = \{\}$;
for i from 1 to d **do**
  for all Partition $P \in bestPA$ **do**
    $\mathbb{G}'_i = copy(\mathbb{G}_i)$ {we will modify some edges' labels}
    for all $(s_i, s_j) \in S^2 | s_i, s_j$ are vertexes in the same P class **do**
      for all $G'_C \in \mathbb{G}'_i$ **do**
        $vc =$ label of edge (s_i, s_j);
        edge (s_i, s_j) becomes labelled by *always*;
        if $vc \in V^C_{far}$ **then**
          erase all edges of G'_C labelled with a comparison value $\geq vc$;
        end if
      end for
    end for
    for all s_i, s_j vertexes such that they are in enemy classes of P for $\bigcup \mathbb{G}_j | j < i$ **do**
      for all $G'_C \in \mathbb{G}'_i$ **do**
        $vc =$ label of edge (s_i, s_j);
        edge (s_i, s_j) becomes labelled by *jamais*;
        if $vc \in V^C_{proche}$ **then**
          erase all edges of G'_C labelled with a comparison value $\leq vc$;
        end if
      end for
    end for
    add best reference partitions for $\mathbb{G}'_i\}$ to $bestPB$;
  end for
  $bestPA = bestPB$;
  $bestPB = \{\}$;
end for
return set of partition values of partitions in $bestPA$;

8.2.2 Improving the interface for librarian users

Finally, there are questions about how to develop and adapt the interface to enable visualization of large Sudoc subsets. We propose two points:

- The first point is based on the fact that a contextual entity represents an edition of a work and that librarians are more interested in works (for example, "*Antigone*" from "SOPHOCLES") than editions of work (two of the "*Antigone*" editions published in "2012" and in "1568"). So, the authority notices representing the editions of the same work from the point of view of the same contributor could be represented together with a special icon representing the work instead of its editions. The details about the work' editions are available when pointing the mouse cursor on the icon representing the work.

- The second point is to focus on the bibliographies of one or several authority notices in a window distinct from the main window with all the data of the considered Sudoc subset. Once it is done, the librarian can come back to the main window.

Chapter 9

Bibliography

[1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23, 2008.

[2] Arvind Arasu, Christopher Ré, and Dan Suciu. Large-scale deduplication with constraints using dedupalog. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 952–963, 2009.

[3] Jean-François Baget, Michel Chein, Madalina Croitoru, Alain Gutierrez, Michel Leclere, and Marie-Laure Mugnier. Logical, graph based knowledge representation with cogui. In *Atelier GAOC: Graphes et Appariement d'Objets Complexes en conjonction avec EGC'10*, pages 15–25, 2010.

[4] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Machine Learning*, volume 56, pages 89–113. Springer, 2004.

[5] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Whang, and Jennifer Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18:255–276, 2009.

[6] Krishna Bharat and Monika R Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111. ACM, 1998.

[7] Indrajit Bhattacharya and Lise Getoor. Entity resolution in graphs. *Mining graph data*, page 311, 2006.

[8] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[9] Mikhail Bilenko, Sugato Basu, and Mehran Sahami. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[10] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE, 2006.

[11] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.

[12] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, pages 1170–1182, 1987.

[13] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. An entity name system (ens) for the semantic web. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 258–272, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[15] Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*, 2012.

[16] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *ACM SIGMOD Record*, volume 27, pages 307–318. ACM, 1998.

[17] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1):65–74, 1998.

[18] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Efficient fuzzy match for evaluating data records, November 13 2007. US Patent 7,296,011.

[19] Surajit Chaudhuri, Venkatesh Ganti, and Rajeev Motwani. Robust identification of fuzzy duplicates. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 865–876. IEEE, 2005.

[20] Michel Chein, Michel Leclère, and Yann Nicolas. Sudocad: A knowledge-based system for object identification. Technical report, LIRMM, INRIA Sophia Antipolis, December 2012.

[21] Michel Chein, Michel Leclère, and Yann Nicolas. Sudocad: A knowledge-based system for the author linkage problem. In *Knowledge and Systems Engineering*, pages 65–83. Springer, 2014.

[22] William W Cohen, Henry Kautz, and David McAllester. Hardening soft information sources. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 255–259. ACM, 2000.

[23] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

[24] Madalina Croitoru, Léa Guizol, and Michel Leclère. On link validity in bibliographic knowledge bases. In *14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'2012)*, volume Advances on Computational Intelligence, pages 380–389, Catania, Italie, July 2012. Springer.

[25] Bruno Paiva Lima Da Silva, Jean-François Baget, and Madalina Croitoru. A generic platform for ontological query answering. In *Research and Development in Intelligent Systems XXIX*, pages 151–164. Springer, 2012.

[26] Anish Das Sarma, Ankur Jain, Ashwin Machanavajjhala, and Philip Bohannon. An automatic blocking mechanism for large-scale de-duplication tasks. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1055–1064. ACM, 2012.

[27] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 85–96, New York, NY, USA, 2005. ACM.

[28] Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 164–172. Association for Computational Linguistics, 2009.

[29] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[30] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[31] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7:3–12, December 2005.

[32] A. Guénoche. Partitions optimisées selon différents critères: évaluation et comparaison. *Mathématiques et sciences humaines. Mathematics and social sciences*, (161), 2003.

[33] Alain Guénoche and Henri Garreta. Representation and evaluation of partitions. In *Classification, Clustering, and Data Analysis*, pages 131–138. Springer, 2002.

[34] Sudipto Guha, Nick Koudas, Amit Marathe, and Divesh Srivastava. Merging the results of approximate match operations. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 636–647. VLDB Endowment, 2004.

[35] Léa Guizol. Agrégation pour la réparation de liens. *25ème Journées francophones d'Ingńierie des Connaissances (IC2014)*, 2014.

[36] Lea Guizol and Madalina Croitoru. Investigating the quality of a bibliographic knowledge base using partitioning semantics. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2014)*, pages 948–955. IEEE, 2014.

[37] Léa Guizol, Madalina Croitoru, and Michel Leclère. On link validity and entity resolution. Technical Report RR-11010, LIRMM, INRIA Sophia Antipolis, http://www.lirmm.fr/ guizol/LVER11010.pdf, 2011.

[38] Léa Guizol, Madalina Croitoru, and Michel Leclère. Aggregation semantics for link validity. *Research and Development in Intelligent Systems XXX (SGAI 2013)*, pages 359–372, 2013.

[39] Léa Guizol, Madalina Croitoru, and Michel Leclère. Aggregation semantics for link validity: technical report. Technical report, LIRMM, INRIA Sophia Antipolis, http://www.lirmm.fr/~guizol/AggregationSemanticsforLinkValidity-RR.pdf, 2013.

[40] Léa Guizol, Olivier Rousseaux, Madalina Croitoru, Yann Nicolas, and Aline Le Provost. An analysis of the sudoc bibliographic knowledge base from a link validity viewpoint. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2014)*, pages 204–213. Springer, 2014.

[41] Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2(1):289–300, 2009.

[42] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[43] Mauricio A Hernández and Salvatore J Stolfo. The merge/purge problem for large databases. In *ACM SIGMOD Record*, volume 24, pages 127–138. ACM, 1995.

[44] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[45] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[46] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM, 2003.

[47] Min-Yen Kan and Yee Fan Tan. Record matching in digital library metadata. *Commun. ACM*, 51:91–94, February 2008.

[48] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[49] Xiangnan Kong and Philip S Yu. Semi-supervised feature selection for graph classification. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 793–802. ACM, 2010.

[50] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[51] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 313–320. IEEE, 2001.

[52] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[53] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[54] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[55] Bing Liu. Web data mining. Springer, 2007.

[56] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, New York, NY, USA, 2000. ACM.

[57] Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.

[58] Matthew Michelson and Craig A Knoblock. Learning blocking schemes for record linkage. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 440. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[59] Jennifer Neville and David Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

[60] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[61] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 903–910. Citeseer, 2001.

[62] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–266. ACM, 2001.

[63] Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 264–271. ACM, 2000.

[64] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[65] Byung-Won On, Ergin Elmacioglu, Dongwon Lee, Jaewoo Kang, and Jian Pei. Improving grouped-entity resolution using quasi-cliques. *Data Mining, IEEE International Conference on*, 0:1008–1015, 2006.

[66] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[67] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the conference on empirical methods in natural language processing*, pages 650–659. Association for Computational Linguistics, 2008.

[68] Vibhor Rastogi, Nilesh Dalvi, and Minos Garofalakis. Large-scale collective entity matching. *Proceedings of the VLDB Endowment*, 4(4):208–218, 2011.

[69] Stuart Russell. Identity uncertainty. In *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, volume 2, pages 1056–1061. IEEE, 2001.

[70] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. Reconciliation de references : une approche logique adaptee aux grands volumes de donnees. In *EGC*, pages 623–634, 2007.

[71] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278. ACM, 2002.

[72] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 572–582, Washington, DC, USA, 2006. IEEE Computer Society.

[73] Neil R. Smalheiser and Vetle I. Torvik. *Annual Review of Information Science and Technology (ARIST)*, volume 43, chapter Author Name Disambiguation. Information Today, Inc, 2009.

[74] Padhraic Smyth and Scott White. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 5th SIAM International Conference on Data Mining*, pages 76–84, 2005.

[75] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 485–492. Morgan Kaufmann Publishers Inc., 2002.

[76] Sheila Tejada, Craig A Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[77] Joshua R Tyler, Dennis M Wilkinson, and Bernardo A Huberman. E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, 2005.

[78] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[79] S.Y. Wang. Existence of a pareto equilibrium. *Journal of Optimization Theory and Applications*, 79(2):373–384, 1993.

[80] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[81] William E Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census, 2006.

[82] Stephan Wölger, Christian Hofer, Katharina Siorpaes, Stefan Thaler, Elena Simperl, and Tobias Bürger. Interlinking data - approaches and tools. Technical report, STI Innsbruck, University of Innsbruck, 2011.

Résumé

Nous proposons une approche qualitative pour la résolution d'entités et la réparation de liens dans une base de connaissances bibliographiques. Notre question de recherche est: "**Comment détecter et réparer les liens erronés dans une base de connaissances bibliographiques en utilisant des méthodes qualitatives ?**". L'approche proposée se décompose en deux grandes parties. La première contribution est une sémantique de partitionnement utilisant des critères symboliques et servant à détecter les liens erronés. La seconde contribution est un algorithme réparant les liens erronés. Nous avons implémenté notre approche et proposé une évaluation qualitative et quantitative pour la sémantique de partitionnement ainsi que prouvé les propriétés des algorithmes utilisés pour la réparation de liens.

Abstract

We propose a qualitative entity resolution approach to repair links in a bibliographic knowledge base. Our research question is: "**How to detect and repair erroneous links in a bibliographic knowledge base using qualitative methods?**" The proposed approach is decomposed into two major parts. The first contribution consists in a partitioning semantics using symbolic criteria used in order to detect erroneous links. The second one consists in a repair algorithm restoring link quality. We implemented our approach and proposed qualitative and quantitative evaluation for the partitioning semantics as well as proving certain properties for the repair algorithms.

LIRMM – 161, rue Ada – 34095 Montpellier Cedex 5 – France
