UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Dottorato di Ricerca in Teorie, metodologie e applicazioni avanzate per la comunicazione, l'informatica e la fisica
XI Ciclo - Nuova Serie

Tesi di Dottorato

Memetic Algorithms for Ontology Alignment

Autilia Vitiello

Ph.D. Program Chair
Prof. Giuseppe Persiano

Supervisors
Prof. Vincenzo Loia
Dott. Giovanni Acampora

November 2012

If we knew what it was we were doing,
it would not be called research, would it?
(Albert Einstein)

Acknowledgements

The work of this thesis would not have been possible without the help, encouragement, and support of many people.

First, I would like to thank my supervisor, Prof. Vincenzo Loia, for the possibility to work under his supervision and for allowing me to follow and develop my ideas freely.

Special thanks go to my daily supervisor, Dr. Giovanni Acampora, for his guidance, insightful discussion, encouragement throughout the development of this thesis. The good advice and the useful feedback have been invaluable on both an academic and a personal level. But I am even more grateful to him for having increased my passion on research.

I would also like to thank all my friends, in particular, Federica, for their encouragement during this challenging time and my Ph.D. mates and my office friends for making my stay in Salerno enjoyable.

My warmest thanks go to my family, my parents, Margherita and Cesare, and, my sister, Angela. Without their love and support, I would not have been equipped to take on this challenge. Last but not least, I wish to deeply thank my boyfriend, Francesco, for his constant support, encouragement, understanding, sweetness and love during this time, but, even more, for his belief in me; ending this thesis work would not have been really hard without him.

Abstract

Semantic interoperability represents the capability of two or more systems to meaningfully and accurately interpret the exchanged data so as to produce useful results. It is an essential feature of all distributed and open knowledge based systems designed for both e-government and private businesses, since it enables machine interpretation, inferencing and computable logic. Unfortunately, the task of achieving semantic interoperability is very difficult because it requires that the meanings of any data must be specified in an appropriate detail in order to resolve any potential ambiguity. Currently, the best technology recognized for achieving such level of precision in specification of meaning is represented by ontologies. According to the most frequently referenced definition [60], an *ontology* is an explicit specification of a conceptualization, i.e., the formal specification of the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them [50]. However, different tasks or different points of view lead ontology designers to produce different conceptualizations of the same domain of interest. This means that the subjectivity of the ontology modeling results in the creation of *heterogeneous ontologies* characterized by terminological and conceptual discrepancies. Examples of these discrepancies are the use of different words to name the same concept, the use of the same word to name different concepts, the creation of hierarchies for a specific domain region with different levels of detail and so on. The arising so-called *semantic heterogeneity problem* represents, in turn, an obstacle for achieving semantic interoperability. In order to overcome this problem and really taking advantage of the ontological representation, the most solid solution is to perform a so-called

ontology alignment process or simply *matching*. This process leads two heterogeneous ontologies into a mutual agreement by detecting a set of correspondences, called *alignment*, between semantically related ontology entities [107]. The increasing relevance of performing an ontology alignment process in several domains of application such as knowledge management, information retrieval, medical diagnosis, e-Commerce, knowledge acquisition, search engines, bioinformatics, the emerging Semantic Web and so on, has led to develop in years numerous tools, named *ontology alignment systems* [78][134]. Among all exploited techniques, due to the complex and time-consuming nature of the ontology alignment process, approximate methods have emerged as a successfully methodology for computing sub-optimal alignments [71]. From this point of view, evolutionary optimization methods [13][67] could represent an efficient approach for facing the problem, and, indeed, genetic algorithms have been already applied to solve the ontology alignment problem as shown in [135][90] by reaching acceptable results. However, classical genetic algorithms suffer from some drawbacks such as premature convergence that makes them incapable of searching numerous solutions of the problem area.

Starting from these considerations, this research work investigates an emergent class of evolutionary algorithms, named Memetic Algorithms (MAs), to efficiently face the ontology alignment problem. MAs are population-based search methods which combine genetic algorithms and local refinements. This marriage between global and local search allows keeping high population diversity and reducing the likelihood premature convergence. Several different works demonstrate how MAs converge to high quality solutions more efficiently than their conventional evolutionary counterparts. In detail, the contribution of this thesis is to propose two ontology alignment systems, named *MemeOptiMap* and *MemeMetaMap*, which exploit MAs to produce an ontology alignment by following two different strategies. In particular, *MemeOptiMap* uses MAs to directly solve the ontology alignment problem as a minimum optimization problem. Instead,

MemeMetaMap follows a meta-optimization approach by using MAs to tune the parameters necessary for performing an ontology alignment process. During the evaluation phase, both systems have been compared with the state of the art by means of a statistical multiple comparison procedure. The test results show that both approaches are competitive, and, in particular, *MemeMetaMap* improves the capabilities of the current ontology alignment processes by working regardless of the user involvement, data availability and the need of a priori knowledge about ontology features, and, yielding high performance in terms of alignment quality with respect to top-performers of well-known Ontology Alignment Evaluation Initiative[1] (OAEI), i.e., a coordinated international initiative aimed at providing means to compare and evaluate different ontology alignment systems.

[1] http://oaei.ontologymatching.org/

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we concentrate our attention on semantic interoperability problem which affects interacting systems characterized by a different knowledge interpretation. When systems model their information domain through ontologies, the interoperability problem is reduced to the ontology alignment problem. The principal aim of this thesis is to face this problem by studying approaches based on an emergent class of evolutionary algorithms, known as Memetic Algorithms. In this chapter, we give an overview and the motivations for our research work (see section 2.1), our contribution (see section 2.2) and the general thesis organization (see section 2.3).

1.1 Overview and Motivation

Interoperability is an essential feature of all distributed and open knowledge based systems designed for both e-government and private businesses. The term interoperability has a broad meaning containing within it many of the issues of effectiveness with which diverse information resources might fruitfully co-exist in common. In particular, in the Information and Communication Technology (ICT) society, the term interoperability represents the ability of two or more systems (which may include organizations, applications, or components) to exchange information and to render useful this information[1]. In practice, it is possible to

[1]Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

distinguish between two levels of interoperability: syntactic and semantic. In detail, if two or more systems are capable of communicating and exchanging data, they are exhibiting *syntactic interoperability*. This level of interoperability involves a common data format and common protocol such as XML or the SQL standards to define any data so that the manner of processing the information will be interpretable from the structure. However, once the syntactical correctness has been verified, the intended meaning of the content of a communication still cannot be judged without some commonality in methods and procedures that each system is employing for modeling it. The ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results represents the *semantic interoperability*. To achieve semantic interoperability, the meanings of any information must be specified in sufficient detail to resolve any potential ambiguity. This requires that both sides must refer to a common and formal information exchange reference model. The current best technology for achieving such level of precision in specification of meaning is represented by ontologies. According to the most frequently referenced definition[60], an *ontology* is an explicit specification of a conceptualization, i.e., the formal specification of the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them [50]. Unfortunately, different tasks or different points of view lead ontology designers to produce different conceptualizations of the same domain of interest. This means that the subjectivity of the ontology modeling results in the creation of *heterogeneous ontologies* characterized by terminological and conceptual discrepancies. Examples of these discrepancies are the use of different words to name the same concept, the use of the same word to name different concepts, the creation of hierarchies for a specific domain region with different levels of detail and so on. The arising so-called *semantic heterogeneity problem* represents, in turn, an obstacle for achieving semantic interoperability. In order to overcome this problem, a simple solution could be that all communicating systems agree on using the same ontology. However, this solution is neither always possible nor desirable. For example, in open environments composed of heterogeneous agents and characterized by the absence of a central control, the agreement on utilizing the same ontology represents a case unfeasible since it could strongly limit the fundamental feature

of flexibility. Moreover, in enterprise scenarios, a typical attitude of the parts is the refusal to convert all the content of their ontologies in a target ontology that is less expressive or not considered as a *de facto* standard [45]. In both scenarios, the semantic heterogeneity problem could be overcome having a single standard ontology containing representations of every term used in every application, but, because of the rapid evolution of language (e.g. creation of new terms or assignments of new meanings to old terms), the creation of a such ontology is generally considered an impossible task.

Therefore, since in short the scenario where two systems exploit the same ontology is almost impossible, the most solid solution for enabling semantic interoperability (even if with a lesser degree than using a common ontology) and really taking advantage of the ontological representation is to perform a so-called *mapping process* which allows overcoming the several forms of heterogeneity which exist between two ontologies. The recent researches about ontologies are heavily focused in the development of different mapping processes, and in particular, the so-called *ontology alignment process* or simply *matching* able to automatically map the definitions used by one system to those of another by producing a so-called *alignment*. Due to the relevance of automatically performing an ontology alignment process in several domains of application such as knowledge management, information retrieval, medical diagnosis, e-Commerce, knowledge acquisition, search engines, bioinformatics, the emerging Semantic Web and so on, in years, numerous tools, named *ontology alignment systems*, have been developed [78][134]. However, in spite of these research efforts, currently, there is no integrated solution that is a clear success, which is robust enough to be the basis for future development, and which is usable by non expert users [117].

By summarizing, computing an alignment is a crucial step for achieving semantic interoperability in several domains of application and with a lot of still open questions. Therefore, addressing the ontology alignment problem is a really world-wide interesting research.

1.2 Contribution

The ontology alignment process is a necessary step for allowing semantic interoperability within distributed systems and web applications. Therefore, designing an efficient ontology alignment process has a crucial relevance in our competitive world. In detail, it consists in identifying a collection of similar entities existing between different ontologies so to lead them in a *semantic reconciliation*. In last years, because of the complex and time-consuming nature of this process, particularly when the considered ontologies are characterized by a significant number of entities, approximate methods have been widely used for computing a sub-optimal ontology alignment [71]. From this point of view, evolutionary optimization methods [13][67] could represent an efficient approach for facing the problem of how to find semantic correspondences between ontologies. As a matter of fact, evolutionary methods such as genetic algorithms have been already applied to solve the ontology alignment problem as shown in [135][90] by achieving acceptable results. However, the inherent issue of premature convergence characterizing classical genetic algorithms makes them incapable of searching numerous solutions of the problem area. Starting from these considerations, this research work investigates an emergent class of evolutionary algorithms, named Memetic Algorithms (MAs), to face the ontology alignment problem and efficiently produce an alignment. MAs are population-based search methods which combine genetic algorithms and local refinements. This marriage between global and local search allows keeping high population diversity and reduce the likelihood premature convergence. Indeed, several different works demonstrate how MAs converge to high quality solutions more efficiently than their conventional evolutionary counterparts. Hence, the choice of exploring this new class of algorithms. In detail, the contribution of this thesis is to propose two ontology alignment systems, named *MemeOptiMap* and *MemeMetaMap*, which exploit MAs for addressing ontology alignment problem by following two different point of views. In particular, *MemeOptiMap* uses MAs to directly solve the ontology alignment problem as a minimum optimization problem. Instead, *MemeMetaMap* exploits MAs to address meta-matching problem, i.e., the issue related to determining the appropriate values for ontology alignment process parameters and, consequently,

it produces an alignment by performing a typical matching characterized by the computed parameters. Both approaches have been compared with the state of art by means of a statistical multiple comparison procedure. The test results show that both approaches are competitive. However, in particular, *MemeMetaMap* improves the capabilities of the current ontology alignment processes by working regardless of the user involvement, data availability and the need of a priori knowledge about ontology features, and, yielding high performance in terms of alignment quality with respect to top-performers of well-known Ontology Alignment Evaluation Initiative campaigns[1].

1.3 Thesis Organization

Before we go further in describing our research work, this section gives the reader a brief overview of the entire thesis:

- *Chapter 2* presents an overview of the ontology alignment domain starting with a brief description of ontologies and their limitations linked to the semantic heterogeneity problem. It contains the formal definition of the ontology alignment process and it presents the state of the art about the systems developed for implementing it. It ends by describing the current challenges characterizing the ontology alignment problem scenario, giving more attention to those addressed by this research work.

- *Chapter 3* introduces the so-called Memetic Algorithms (MAs) which represent the main methodology used in our research work for facing the ontology alignment problem. The chapter presents the general structure of MAs, starting with basic concepts about search algorithms and the main components combined in a MA template represented by local search methods and genetic algorithms. Then, a description of one of the extensions existing of MAs represented by parallel memetic algorithms is given. The chapter ends with a discussion about the techniques used to execute a performance comparison among MAs, and, in general, among different approaches.

[1] http://oaei.ontologymatching.org/

- *Chapter 4* accurately describes our first contribution to ontology alignment consisting in designing and implementing a memetic algorithm-based ontology alignment system named *MemeOptiMap*. The chapter includes the formulation of the ontology alignment problem as an optimization one and the complete description of all components of system *MemeOptiMap*. However, since implementing an efficient memetic algorithm requires to chose a suitable configuration of parameters, the chapter presents also the research work aimed at investigating different settings to find the best local configuration for *MemeOptiMap* in a multi-agent system scenario. The chapter ends by describing a parallel version of the designed system *MemeOptiMap* aimed at reducing computational cost.

- *Chapter 5* discusses our second contribution consisting in producing satisfactory alignments by addressing the ontology meta-matching problem. In detail, the chapter describes all details related to the implementation of a meta-matching system, named *MemeMetaMap*, which exploits a memetic algorithm for optimizing the selection of the best ontology alignment parameters (weights and threshold). The chapter ends by describing a fuzzy logic-based improvement designed for managing some specific instance parameters affecting *MemeMetaMap*'s behaviour.

- *Chapter 6* is aimed at presenting the comparison between our ontology alignment systems based on memetic algorithms and the existing ones in literature. This comparison is performed through a statistical multiple comparison procedure on a well-known dataset provided by the Ontology Alignment Evaluation Initiative.

- *Chapter 7* presents conclusions by summarizing strengths of our ontology alignment systems but also weaknesses to be addressed with future works.

Chapter 2

The Ontology Alignment Problem

In this chapter, we introduce the basic concepts concerning with the main topic of this thesis: the ontology alignment problem. We start with a brief description of ontologies (see section 2.1) and their limitations due to the semantic heterogeneity problem (see section 2.2). Then, we give the formal definition of an ontology alignment process (see section 2.3) and present the state of the art about the systems which have been developed so far for implementing it (see section 2.4). We conclude by describing the current challenges characterizing the ontology alignment problem scenario, giving more attention to those addressed by this research work (see section 2.5).

2.1 Ontologies

The word "ontology" is a rather overloaded term, which is used with several different meanings in different communities [62]. Perhaps, the most ancient sense concerns with the philosophical discipline which considers the "Ontology" as the study of the *nature* and *structure* of the things *per se*. In particular, in his Metaphysics[1], Aristotle defined the Ontology as the science of "being *qua* being", i.e., the study of attributes that belong to anything just because of its existence. How-

[1] http://classics.mit.edu/Aristotle/metaphysics.html

ever, in this thesis, we focus on the computational sense of an ontology, i.e., the notion of ontologies from a Computer Science vision. In detail, Computer Science gives a meaning to the word "ontology" starting from a point of view completely different from Philosophy. Indeed, according to [61], "For knowledge-based systems, what "exists" is exactly that which can be represented". Therefore, computational ontologies are viewed as means to formally model the structure of a system, i.e., the relevant entities and relations that emerge from its observation, and which are useful to achieve prefixed purposes [62].

Precisely, the term "ontology" was introduced to the information sciences during the 1990s by several Artificial Intelligence (AI) research communities. In particular, one of the first definitions of a computational ontology (from now referred only as ontology) was coined by Gruber [60] in the 1993 as follows: "an ontology is an explicit specification of a conceptualization". More in detail, an ontology explicitly provides a specification of a conceptualization viewed as the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [51].

A few years later, Borst [17] redefined an ontology as "a formal specification of a shared conceptualization". The adjective "formal" means that the ontology specification must be expressed in a formal language characterized by a specific syntax and semantics so as to result in a machine executable and machine interpretable representation of the world. Hence, a lot of languages have been developed for allowing an effective use of ontologies (see section 2.1.2). Regarding the adjective "shared", it aims at capturing the aspect that an ontology should express a world view that reaches the consensus of several parts. The reason which prompted Borst to include this feature was that the ability to reuse an ontology is almost null if the conceptualization it defines is not generally approved. Indeed, just the introduction of this capability to ontologies has allowed their great diffusion as the most important means for exchanging and reuse of information in all modern knowledge based systems (see section 2.1.4). However, the practical usage of ontologies and their usefulness in the sharing information can be limited by the so-called *ontology heterogeneous problem* that will be described in section 2.2.

Finally, in 1998, Studer et al. [122] joined the previous two definitions result-

ing in the following and nowadays most frequently seen definition: "An ontology is a formal, explicit specification of a shared conceptualization". However, the aforementioned definition is assumed to be informal. Section 2.1.1 presents a mathematical definition of an ontology together with a description of all its basic components, whereas, for an ontology example see section 2.1.3.

In literature, there are described two main classes of ontologies: the former explicitly captures "static knowledge" about a domain, in contrast to the latter that provides a reasoning point of view about the domain knowledge (problem solving knowledge) [122]. In turn, in these two classes, it is possible to distinguish other subclasses. In particular, in the first class, a distinction between types is made on the basis of the level of generality, as summarized below [122]:

- *Domain ontologies* capture the knowledge valid for a particular type of domain (e.g. electronic, medical, mechanic, digital domain);
- *Generic ontologies* are valid across several domains (examples of this kind of ontology are SUMO[1] and DOLCE[2]);
- *Application ontologies* contain all the necessary knowledge for modelling a particular domain (usually a combination of domain and method ontologies) [52].
- *Representational ontologies* do not commit to any particular domain. Such ontologies provide representational entities without stating what should be represented. A well-known representational ontology is the Frame Ontology [61], which defines concepts such as frames, slots and slot constraints allowing to express knowledge in an object-oriented or frame-based way.

Instead, the second class can be divided in:

- *Task ontologies* provide terms specific for particular tasks (e.g. hypothesis belongs to the diagnosis task ontology);
- *Method ontologies* provide terms specific to particular Problem-Solving-Methods [52] (e.g. correct state belongs to the Propose-and-Revise method ontology).

[1] http://www.ontologyportal.org/
[2] 3http://www.loa-cnr.it/DOLCE.html

In this thesis, in particular, we deal with domain ontologies named simply ontologies.

2.1.1 Basic components of an ontology: formal description

In literature, there are a lot of formal definitions of an ontology [36][43][120]. Typically, a definition is designed for fitting the needs and goals of the researcher. According to [61], the key components of an ontology are the following ones:

- *class* or *concept*: it represents the abstract view of a set of objects which share common features in the domain of the interest. For instance, "student" could be a class which represents all students of a college;

- *property* or *relation*: it allows to express relationships between two concepts in a given domain of interest. More precisely, it describes the relationship between the first concept, represented in the *domain*, and the second one, represented in the *range*. For instance, "study" could be represented as a relationship between the concept "student" (which is a concept in the domain) and "subject" (which is a concept in the range);

- *individual* or *instance*: it is the "ground-level" component of an ontology. Indeed, it represents a specific world object corresponding to a class. For instance, "Math" could be an instance of the class "subject";

- *function*: it represents a special case of relations, in which the last object in each tuple is unique given the preceding objects. In other words, a function of N arguments is a relation of $N + 1$ arguments in which the value of the last argument is a function of the first N arguments. For instance, kinds of functions are the *specialization* which allows to link a class to its superclasses or the *instantiation* which allows to relate an instance with its class. Examples of these functions are: the concept "student" could be a specialization of the concept "person", whereas, "Francesco" could be an instance of the concept "student";

- *axiom*: it allows to explicitly express propositions that are always true. It can be used to verify the consistency of the ontology or to infer new information.

Later, Ehrig [36] gives a formal definition of an ontology by organizing it in a modular way. Indeed, he defines classes, properties and the specialization functions, respectively, among classes and properties as the *Core Ontology*, whereas, he considers instances and the instantiation function as part of a separate *knowledge Base*. In this way, Ehrig divides the intentional aspects of a domain, enclosed in the core ontology, from the extensional part, provided by a knowledge base. On the contrary, in the same period, Euzenat and Shvaiko [43] join the intensional and extensional components of an ontology giving a definition containing at the same level each one of the aforementioned components. In addition, apart from specialization and instantiation functions, Euzenat and Shvaiko explicitly define other functions, such as disjointness (exclusion) and assignment.

In this thesis, we take a cue from all aforementioned formal approaches for giving a mathematical definition of an ontology as follows.

Definition 1 (Ontology) *An ontology is a 9-tuple* $O =< C, P, I, A, \leq_C, \leq_P, \phi_{CP}, \phi_{CI}, \phi_{PI} >$ *such that:*

- C *is a nonempty set of classes;*
- P *is a nonempty set of properties;*
- I *is a set of instances (it can be empty);*
- A *is a set of axioms, preferably nonempty;*
- $\leq_C$ *is a partial order on* C*, called class hierarchy or taxonomy;*
- $\leq_P$ *is a partial order on* R*, called property hierarchy;*
- $\phi_{CP} : P \rightarrow C \times C$ *is a function which associates a property* $p \in P$ *with two classes linked between them just through the relation* p*. We denote with* domain $dom(p) := \pi_1(\phi_{CP}(p))$ *and* range $ran(p) := \pi_2(\phi_{CP}(p))$*;*

- $\phi_{CI} : C \rightarrow \mathcal{P}(I)$ *is a function which associates a concept* $c \in C$ *with a subset of* I *representing the instances of the concept* c*;*

- $\phi_{PI} : P \rightarrow \mathcal{P}(I^2)$ *is a function which associates a property* $p \in P$ *with a subset of cartesian product* $I \times I$ *representing the pair of instances related through the property* p*.*

Similar to Ehrig's definition, we denote that if $c_1 \leq_C c_2$, where $c_1, c_2 \in C$, then c_1 is a subclass of c_2 and c_2 is a superclass of c_1. At the same way, if $p_1 \leq_P p_2$, where $p_1, p_2 \in P$, then p_1 is a subproperty of p_2 and p_2 is a superproperty of p_1. Besides, we also borrow, in part from Ehrig, the term *entity* to denote a class, a property or an instance in a generic way. In other words, an entity e in an ontology verifies this constraint: $e \in C \cup P \cup I$.

2.1.2 Languages for ontologies

Ontologies can be encoded by means of many languages that range from hardly to highly formalized. In particular, different requirements should be satisfied when the design of an ontology language is addressed. In particular, an ontology language should have:

- a well-defined syntax;
- a well-defined semantics;
- an efficient reasoning support;
- a sufficient expressive power;
- a convenience of expression.

Since, unfortunately, the more powerful language for expressing the facts, the higher computational costs, according their purposes, designers of ontology are constrained to choose language with regard to the trade-off between *expressiveness* and *efficiency* [123]. This constrain leads to the consideration of different degrees of formalism for ontology languages [129]:

- *Highly informal*: if ontologies are expressed in natural language. According to this, a highly informal ontology would not be an ontology, since it is not machine-readable;
- *semi-informal*: if ontologies are expressed in a restricted and structured form of natural language;
- *semi-formal*: if ontologies are expressed in an artificial and formally defined language (e.g. RDF graphs);
- *Rigorously formal*: if the ontology languages provide meticulously defined terms with formal semantics, theorems and proof of properties such as soundness and completeness (e.g. Web Ontology Language [OWL]).

Although the informal languages have the advantage to produce simpler and faster representation, in order to make an ontology machine executable and interpretable a formal language is necessary. The first formal languages used to express an ontology derives from the knowledge representation[1] (KR) subfield of AI. The most popular languages in the group of KR languages used for ontology encoding are description logics (DL), i.e., subsets of first-order logic. Instead, one of the first ontology-dedicated languages was Stanford's Ontolingua [59] whose the syntax and semantics are based on a standard notation for predicate calculus called Knowledge Interchange Format[2] (KIF). Since then, the area of ontology dedicated languages has grown more and more. Relevant examples are Ontology Interface Layer (OIL) [44] and XML-based Ontology Exchange Language[3] (XOL). However, because of the enormous development of the Web, currently, the most usual ontology language is the Web Ontology Language[4] (OWL). In detail, OWL is an ontology language developed by the World Wide Web Consortium (W3C) Web Ontology Working Group[5] as part of its Semantic Web activity. The development of OWL was motivated by the key role foreseen for ontologies

[1]Knowledge representation is an area of AI research aimed at representing knowledge in symbols to facilitate inferencing from those knowledge elements, creating new elements of knowledge.

[2]http://www.ksl.stanford.edu/knowledge-sharing/kif/

[3]http://xml.coverpages.org/xol.html

[4]http://www.w3.org/TR/owl-guide/

[5]http://www.w3.org/2001/sw/WebOnt/charter

in the Semantic Web (i.e., providing precisely defined and machine processable vocabularies that can be used in semantically meaningful annotations), and the recognition that existing web languages, such as RDF, were not expressive enough for this task [58]. Precisely, OWL has features from several families of representation languages, including primarily Description Logics. It is defined as three sublanguages aimed at fulfilling the ontology language requirements that, as described above, are in contrast among them. In particular, OWL sublanguages are [12]:

- *OWL Full*: it represents the entire language composed by all the OWL languages primitives. It also allows to combine these primitives in arbitrary ways with RDF. The disadvantage of OWL Full is the language has become so powerful as to be undecidable, dashing any hope of complete reasoning support;
- *OWL DL*: it restricts the way in which the constructors from OWL and RDF can be used in order to regain computational efficiency. Therefore, the advantage of this is that it permits efficient reasoning support, whereas, the disadvantage is that we loose full compatibility with RDF;
- *OWL Lite*: it is an ever further restriction limited. The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is of course a restricted expressivity.

A more exhaustive overview about ontology languages can be found in [56]. In this thesis, the used ontologies are encoded in OWL language.

2.1.3 An ontology example

To better illustrate the meaning of an ontology used in this thesis, we present an ontology example which could be related to a stock of a car dealer [36]. The example has six classes (*object*, *vehicle*, *owner*, *boat*, *car*, *speed*), two properties indicating the belonging to somebody and speed, three individuals (*Paul*, *Fiat 500*, *160 km/h*). There are three specialization relations (between *object* and *vehicle*, between *vehicle* and *boat*, between *vehicle* and *car*). Each vehicle has an

owner and each car has a fixed speed. On instance level, the Fiat 500 belongs to Paul and is characterized by the speed of 160 km/h. Furthermore, there is an axiom: every car needs to have at least one owner. Formally, this ontology is defined as $O_d =< C, P, I, A, \leq_C, \leq_P, \phi_{CP}, \phi_{CI}, \phi_{PI} >$ where:

- $C = \{object, vehicle, owner, boat, car, speed\}$;
- $P = \{belongsTo, hasSpeed\}$;
- $I = \{Paul, Fiat500, 160km/h\}$;
- $A = \{\forall x \exists y : car(x) \Rightarrow belongsTo(x, y)\}$;
- $\leq_C = \{(vehicle, object), (boat, vehicle), (car, vehicle)\}$;
- $\leq_P = \emptyset$;
- $\phi_{CP} = \{belongsTo \rightarrow (vehicle, owner), hasSpeed \rightarrow (car, speed)\}$;
- $\phi_{CI} = \{owner \rightarrow Paul, car \rightarrow Fiat500, speed \rightarrow 160km/h\}$;
- $\phi_{PI} = \{belongsTo \rightarrow \{(Fiat500, Paul)\}, hasSpeed \rightarrow \{(Fiat500, 160km/h)\}\}$.

Hoverer, in order to give a more immediate reading of an ontology, in this thesis, we use also a visual language whose the graphical notation is presented in Fig. 2.1. In detail, classes are depicted as rectangular boxes, properties as hexagons, and individuals as rounded boxes. Specialization relations are drawn as solid arrows. A relation has an incoming arrow from its domain and an outgoing arrow to its range. The instantiations of concepts and relations are depicted as dotted, arrowed lines. Fig. 2.2 shows the ontology O_d in a graphical way.

Finally, as above said, in this thesis, the used language for producing a machine executable version of considered ontologies is OWL. Therefore, it seems useful to conclude our example with a fragment (see listing 2.1) of the OWL representation of the ontology O_d. For readability, the namespaces of RDF resources are abbreviated. After the namespace declaration, a class "auto#vehicle" is defined. This class has the English label "vehicle". Further, it is defined to be a subclass of "auto#object" before of the closing class-tag. The other entities have some more tags, but are built-up accordingly. The axiom is represented as "owl:Restriction".

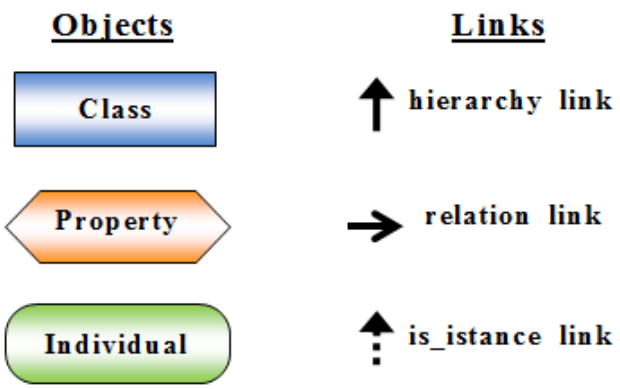


Figure 2.1: Visual notations

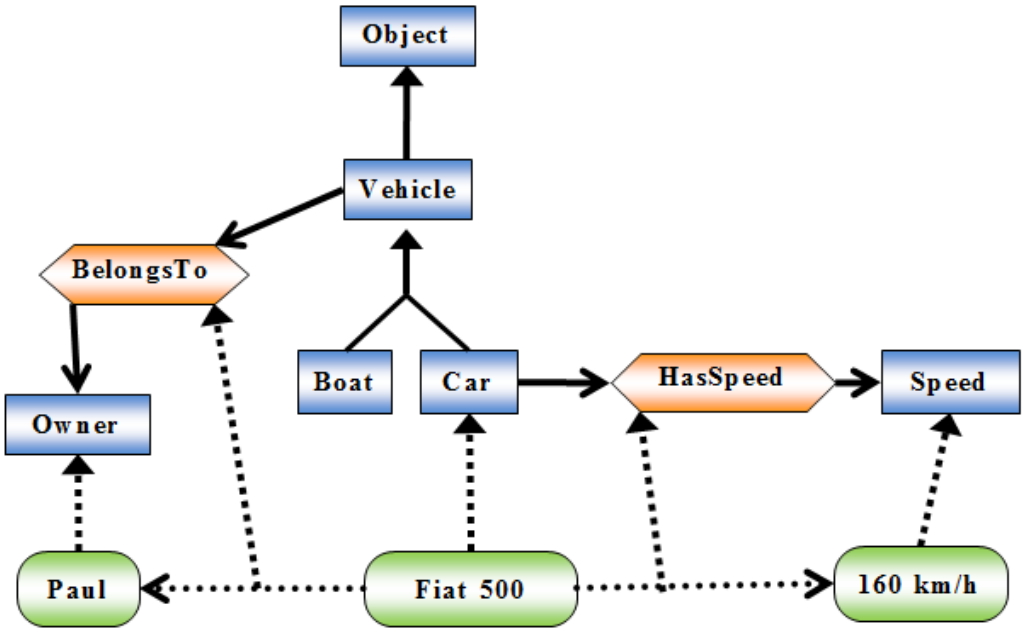


Figure 2.2: A graphical representation of an ontology related to a stock of a car dealer.

```
<rdf:RDF
...
xmlns:auto="http://www.aifb.uni-karlsruhe.de/WBS/meh/
auto1.owl">
   <owl:Class rdf:about='auto#vehicle'>
      <rdfs:label xml:lang='en'>vehicle</rdfs:label>
      <rdfs:subClassOf rdf:resource='auto#object'/>
   </owl:Class>
   <owl:Class rdf:about='auto#car'>
      <rdfs:label xml:lang='en'>car</rdfs:label>
      <rdfs:subClassOf rdf:resource='auto#vehicle'/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource='auto#belongsTo'/>
            <owl:minCardinality>1</owl:minCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>
    ........
   <owl:ObjectProperty rdf:about='auto#belongsTo'>
      <rdfs:label xml:lang='en'>belongs to</rdfs:label>
      <rdfs:domain rdf:resource ='auto#vehicle'/>
      <rdfs:range rdf:resource='auto#owner'/>
   </owl:ObjectProperty>
   ........
   <auto:Owner rdf:about='auto#Paul'/>
   ........
   <auto:Car rdf:about='auto#Fiat500'>
      <rdfs:label xml:lang='en'>Fiat 500</rdfs:label>
      <auto:belongsTo rdf:resource='auto#Paul'/>
      <auto:hasSpeed rdf:resource='auto#160km/h'/>
   </auto:Car>
</rdf:RDF>
```

Listing 2.1: OWL fragment representing ontology O_d

2.1.4 The role of ontologies in different fields

We conclude this section about ontologies by giving a description of their enormous applicability. In years, ontologies have been successfully applied in different fields such as knowledge management, naive physics, information retrieval, medi-

cal diagnosis, natural language processing, e-Commerce, information integration, knowledge acquisition, search engines, bioinformatics and the emerging Semantic Web. In particular, in search engine field, ontologies have been used in the form of thesauri to find synonymous of terms in order to facilitate internet searching [106]. In E-commerce, ontologies contribute to communication between seller and buyer thanks to human and machine-readable description of merchandise [44][126]. In naive physics, ontologies have been exploited for formalizing knowledge about physical objects such as liquids [66] and electrial components [85]. Ontologies are also exploited for the development of approaches for the extraction of knowledge from abstracts of scientific articles [132][94] and for supporting systems aimed at acquiring knowledge from domain experts such as Protégé project[1]. Moreover, ontologies have been useful in medical domain thanks to capability of formalizing diagnosis, therapy planning and patient monitoring [14][35]. In addition, several bio-ontologies have been defined such as the Gene Ontology[2] (GO) and Ontology for Molecular Biology[3] (MBO) in order to extend existing taxonomies related to biological knowledge. However, the most relevant use of the ontology is surely in the Semantic Web, whose ontologies represent the backbone. More in detail, the aim of the Semantic Web is to represent information more meaningfully for humans and computers alike. Therefore, Semantic Web enables the description of contents and services in machine-readable form, and enables annotating, discovering, publishing, advertising and composing services to be automated [130]. In this scenario, the task of ontologies is to annotate semantics and provide a common, comprehensible foundation for resources on the Semantic Web.

In short, it is possible to summarize the role of an ontology in different fields as:

- Constituting a community reference;
- Sharing consistent understanding of what information means;
- Making possible knowledge reuse and sharing.

[1] http://protege.stanford.edu/
[2] http://genome-www.stanford.edu/GO/
[3] http://igd.rz-berlin.mpg.de/%CB%9Cwww/oe/mbo.html

However, in spite of their applicability, the role of ontologies in different fields can be limited by the so-called *semantic heterogeneity problem* as described in the next section.

2.2 Semantic heterogeneity problem

In a distributed and open system, such as the semantic web and many other applications presented in the previous section, heterogeneity cannot be avoided. Different actors have different interests and habits, use different tools and knowledge, and most often, at different levels of detail [43]. These various reasons for heterogeneity lead to diverse forms of heterogeneity, the most obvious types of heterogeneity are reported below:

- *Syntactic heterogeneity*: it occurs when two ontologies are not expressed in the same ontology language. This happens, for example, when two ontologies are defined through different knowledge representation formalisms, for instance, OWL and F-logic;
- *Terminological heterogeneity*: it includes all forms of mismatches that are related to the process of naming the entities belonging to ontologies at hand. Typical examples of mismatches at the terminological level are:
  - different words are used to name the same entity (synonymy);
  - the same word is used to name different entities (polysemy);
  - words from different languages (English, French, Italian, Spanish, German, Greek, etc.) are used to name entities;
  - syntactic variations of the same word (different acceptable spellings, abbreviations, use of optional prefixes or suffixes, etc.).
- *Conceptual heterogeneity*: it stands for the differences in modelling the same domain of interest. In other words, it includes mismatches which have to do with the content of an ontology. Discrepancies belonging to this category can be divided in two main classes:

- *metaphysical differences*, which have to do with how the world is "broken into pieces" (i.e., what entities, properties and relations are described in an ontology);
- *epistemic differences*, which have to do with the assertions that are made about the selected entities.

In turn, metaphysical differences depend on three important reasons:

- *Difference in coverage*: it occurs when two ontologies describe different, possibly overlapping, regions of the world at the same level of detail and from a unique perspective. This is, for example, the case for two partially overlapping geographic maps;
- *Difference in granularity*: it occurs when two ontologies describe the same region of the world from the same perspective but at different levels of detail. For example, this happens when two geographic maps model the same region with different scales;
- *Difference in perspective*: it occurs when two ontologies describe the same region of the world, at the same level of detail, but from a different perspective. This occurs, for example, for maps with different purposes: a political map and a geological map do not display the same objects.

- *Semiotic heterogeneity*: it is concerned with how entities are interpreted by people. Indeed, entities which have exactly the same semantic interpretation are often interpreted by humans with regard to the context, for instance, of how they are ultimately used. This kind of heterogeneity is difficult for the computer to detect and even more difficult to solve, because it is out of its reach.

Usually, several types of heterogeneity occur together. In this work, we only concern with the terminological and conceptual types of heterogeneity by denoting them as the so-called *semantic heterogeneity problem*. To provide an example of the semantic heterogeneity problem, Fig. 2.3 shows an ontology which describes the same domain as the ontology depicted in Fig. 2.2, but it uses different terms

(terminological heterogeneity), a different coverage of domain and a different granularity of specialization.

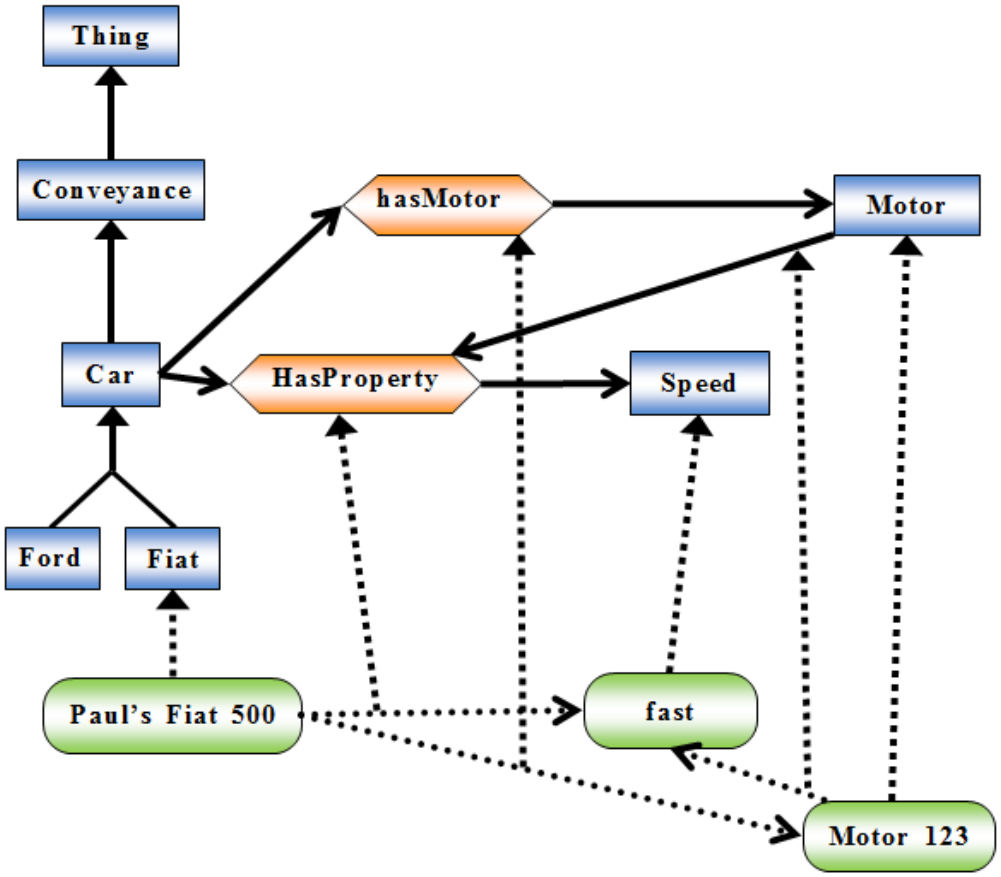


Figure 2.3: Example of ontology

In order to guarantee an appropriate level of interoperability between systems and really take advantage of the ontology benefits, it is necessary to address the semantic heterogeneity problem. Currently, the most solid solution to lead heterogeneous ontologies into mutual agreement is represented by a so-called *mapping process*. The recent researches about ontologies are heavily focused in the development of different mapping processes. In particular, it is possible to distinguish among three kinds of processes:

- *integration*: the process of generating a single ontology about a subject from two or more existing ontologies about different subjects[108];

- *merging*: the process of generating a single, coherent ontology from two or more existing ontologies related to the same subject[108];

- *alignment process* or simply *matching*: the process of detecting correspondences among existing ontologies in order to make them consistent. Differently from the previous methods, the original ontologies are kept separate.

In particular, our work deals with the matching operation. Therefore, a more detailed description of this process is given in the next section.

2.3 Ontology Alignment Process

Nowadays, ontologies are recognized as a fundamental component for enabling interoperability across heterogeneous systems and distributed applications thanks to their capability of formally describing the semantics of a particular domain of interest. However, in spite of their large exploitation, the ability of ontologies to manage disparate information could be limited by the so-called *semantic heterogeneity problem*. This problem is caused by the enormous variety of ways that a domain of interest can be conceptualized which leads to the creation of different ontologies with contradicting or overlapping parts [121]. Therefore, in order to address the semantic heterogeneity problem, a so-called *ontology alignment process* or *matching process* is required. This process aims at detecting a set of correspondences between semantically related entities of different ontologies [107] in order to lead them into a mutual agreement. The set of correspondences is called *alignment*. The importance of performing an ontology alignment process to enable communication and data exchange among people, organizations and software agents is evident in different scenarios (see section 2.3.3). However, aligning two ontologies is a tedious process and manually impractical, especially when involved ontologies are of considerable size (containing hundreds of elements). Hence, the development of numerous tools, named *ontology alignment systems*, capable of performing automatic or semi-automatic ontology alignment process (see section 2.3 for the state of the art). The increasing number of methods available for ontology matching has inspired the creation of a coordinated international initiative, the Ontology Alignment Evaluation Initiative (OAEI), aimed

at providing means to compare and evaluate different ontology alignment systems (see section 2.3.2). In the next section, a formal definition of an ontology alignment process is given, whereas, for an ontology alignment example see section 2.3.4.

2.3.1 Formal definition

An ontology alignment process detects matchings between two ontologies O_1 and O_2 and produces in output a so-called alignment A. However, in its formal definition, the ontology alignment process specifies additional and optional inputs, such as: a partial alignment A' which is to be completed by the process to obtain the output alignment A; some parameters p such as weights and thresholds and some external resources r such as common knowledge or dictionaries. Therefore, according to [43], the ontology alignment process can be formally defined as follows:

Definition 2 (Ontology Alignment Process) *The ontology alignment process can be seen as a function f which, from a pair of ontologies O_1 and O_2 to align, an input alignment A', a set of parameters p, a set of resources r, returns a new alignment A between these ontologies:*

$$A = f(O_1, O_2, A', p, r).$$

Apart from its inputs, an ontology alignment process is strongly dependent on a collection of additional features, known as *matching dimensions*. These dimensions can be considered as a set of constraints affecting the behavior of the alignment process and determining its difficulty. Their complete description is left to the section 2.3.1.1. Graphically, an ontology alignment process can be represented as in Fig. 2.4.

The output alignment A is a set of so-called *mapping elements*. Each mapping element is used for linking an entity belonging to the first ontology with a similar entity belonging to the second ontology. Formally,

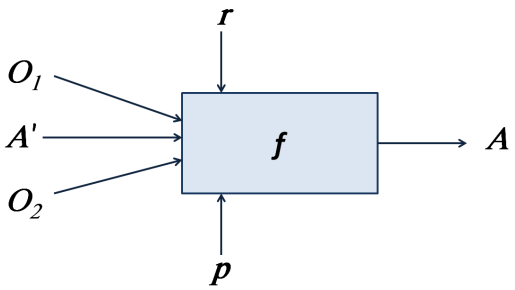


Figure 2.4: The ontology alignment process

Definition 3 (Mapping Element) *A mapping element is a 4-uple:* $\langle e, e', n, R \rangle$ *where*

- e *and* e' *are the entities (e.g., XML elements, properties, classes) of the first and the second ontology respectively;*
- n *represents the* confidence value *(typically in the [0,1] range) which represents the closeness existing between the entities* e *and* e'*;*
- R *is a relation (e.g., equivalence (=); disjointness (⊥); overlapping (⊓)) holding between the entities* e *and* e'*.*

However, since the most typically considered relation is the equivalence, the mapping element can be reduced to a triple $\langle e, e', n \rangle$ where equivalence relation is implicit. We denote this triple simply as *correspondence*.

Only correspondences with a confidence value greater than a given threshold value $t \in [0, 1]$ are considered valid and can be inserted in the output alignment A (*filter operation*). For this reason, the threshold value represents a critical parameter for the ontology alignment process which must be opportunely chosen in order to reduce both the discarding of correct matches and the accepting of wrong ones.

By summarizing, an alignment A between two ontologies O_1 and O_2 can be formally defined as follows:

Definition 4 (Alignment) *An alignment A is a set of k correspondences defined as follows:*

$$A = \{c_l = (e_i, e_j, \eta_l) \ with \quad \begin{array}{l} i \in \{1, 2, \ldots, |O_1|\}, \\ j \in \{1, 2, \ldots, |O_2|\}, \\ \eta_l \in [0, 1], \\ \eta_l > t \in [0, 1], \\ l = 1, 2, \ldots, k\} \end{array}$$

where e_i is the i^{th} entity of ontology O_1, e_j is the j^{th} entity of ontology O_2, η_l is the confidence value of the l^{th} correspondence, t is the threshold value used to filter valid correspondences and the implied relation is the equality.

In order to compute the *confidence value*, which represents the closeness between the two entities composing a correspondence, a so-called *similarity measure*, also known as *matcher*, is used. In literature, there exist different matchers categorized as lexical, linguistic and structural [112][38]. In detail, lexical matchers compute a string distance-based similarity between two entities by taking into account the morphology of the words which characterize them (such as names, comments, etc.); a linguistic matcher determines a similarity value between two entities by taking into account semantic relations such as synonymy and hypernymy; structural matchers compute a similarity value between two entities by considering their kinship (parents and children). See section 2.3.1.2 for a detailed description of some similarity measures in literature. Since the application of a single matcher is often not enough to produce an acceptable output alignment, the common strategy is to combine different matchers to compute a confidence value as an aggregated similarity value [36]. The process of aggregating different similarity measures is commonly known as *similarity aggregation*. Formally, let consider an alignment A, a correspondence c belonging to the alignment A and h similarity measures, the aggregated similarity value for c can be defined as follows:

$$sim_{aggregate}(c) = \sum_{i=1}^{h} w_i \times sim_i(c) \quad \text{subject to} \quad \sum_{i=1}^{h} w_i = 1 \tag{2.1}$$

where w_i is the weight associated to the i^{th} similarity measure and $sim_i(c)$ is the similarity value computed for the correspondence c by the i^{th} similarity measure.

In literature, there are several aggregation strategies, some of them are described in a more detailed way in section 2.3.1.3. Obviously, the quality of the alignments is strongly dependent on selecting of the appropriate similarity measures, weights and thresholds. Therefore, this selection represents a crucial issue in the ontology alignment scenario known as *ontology meta-matching problem* [91]. The techniques which try to solve the ontology alignment problem by addressing the ontology meta-matching problem are referred as *meta-matching systems*.

2.3.1.1 Matching dimensions

Each of the elements featured in definition 2 of ontology alignment process can have specific characteristics which influence the difficulty of the alignment task. It is important to determine and control these characteristics, known as *matching dimensions*, for characterizing the ontology alignment systems known or yet to be invented and then in which situation they are appropriate. We review below some of the most important dimensions as described in [105]:

- *input ontologies*:
  - *heterogeneity of the input languages*: are they described in the same knowledge representation languages? This corresponds to asking for the non emptyness of the syntactic component of the resulting alignment.
  - *languages*: what are the languages of the ontologies (especially in case of homogeneous languages)? Example of languages are KIF, OWL, RDFS, UML, F-Logic, etc.
- *input alignment*:
  - complete/update: Is the alignment process required to complete an existing alignment? (i.e., A is non empty).
  - multiplicity: How many entities of one ontology can correspond to one entity of the others? Usual notations are 1:1, 1:m, n:1 or n:m.

However, for ontology alignment, we prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These assertions could be provided as input (or constraint) for the alignment algorithm or be provided as a result by the same algorithm.

- *input parameters*:
  - *oracles/resources*: Are oracle authorized? If so, which ones (the answer can be any)? Is human input authorized?
  - *training*: Can training be performed on a sample?
  - *proper parameters*: Are some parameters necessary? And what are they? This point is quite important when a method is very sensitive the variation of parameters. A good tuning of these must be available.
- *output alignment*:
  - *multiplicity*: The multiplicity of the output alignment is similar to that of the input alignment (see above).
  - *relations*: Should the relations involved in the correspondences be only equivalence relations or could they be more complex?
- *alignment process*:
  - *resource constraints*: Is there a maximal amount of time or space available for computing the alignment?
  - *Language restrictions*: Is the mapping scope limited to some kind of entities (e.g., only T-box, only classes)?

2.3.1.2 Similarity measures

Similarity measures or matchers are techniques aimed at evaluating the semantic closeness between two ontology entities. In literature, it are categorized in lexical,

linguistic and structural measures. In the sequel, some distance-based similarity measures belonging to these three groups and used in our research work are presented.

In general, **lexical matchers** compute a string distance between a pair of ontology entities by taking into account the morphology of the words which characterize them. In details, by considering an ontology modeled through OWL language, the strings related to ontology entities which can be chosen to compute a distance are:

- the names of ontology entities;
- the labels which could annotate the ontology entities which occur as $rdfs:label$ annotations in OWL representation;
- the comments which could be associated to entities in order to describe them in natural language which are included in an entity through the $rdfs:comment$ annotation in OWL representation.

In literature, there are a lot of string distance methods. Some of these are described below.

The *Levenshtein distance* [84] represents the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character. Mathematically, the levenshtein distance $lev_{a,b}(|a|,|b|)$ between two strings a and b is defined as follows:

$$lev_{a,b}(l_1,l_2) = \begin{cases} 0 & \text{if } l_1 = l_2 = 0 \\ l_1 & \text{if } l_2 = 0 \text{ and } l_1 > 0 \\ l_2 & \text{if } l_1 = 0 \text{ and } l_2 > 0 \\ min \begin{cases} lev_{a,b}(l_1-1,l_2)=1 \\ lev_{a,b}(l_1,l_2-1)+1 \\ lev_{a,b}(l_1-1,l_2-1)+[a_{l_1} \neq b_{l_2}] \end{cases} & \text{otherwise} \end{cases}$$

where the first element in the minimum function corresponds to insertion (from a to b), the second to deletion and the third to match or mismatch, depending on whether the respective symbols are the same.

The *Jaro distance* [75] between two strings a and b is defined as follows:

$$d(a,b) = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3} * (\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}) & \text{otherwise} \end{cases}$$

where m is the number of matching characters and t is half the number of transpositions.

Therefore, Jaro distance takes into account only the commonalities between two entities. Instead, in order to compute the closeness between two entities, the *smoa distance* [121] considers both commonalities and differences characterizing the entities at issue. Formally, the smoa distance between two strings s_1 and s_2 is defined by the following equation:

$$smoa(s_1, s_2) = comm(s_1, s_2) - diff(s_1, s_2) + winklerImpr(s_1, s_2)$$

where $comm(s_1, s_2)$ stands for the commonality between s_1 and s_2, $diff(s_1, s_2)$ for the difference and $winklerImpr(s_1, s_2)$ for the improvement of the result using the method introduced by Winkler in [140]. More in detail, the commonality is computed by means of the substring string metric. In particular, the biggest common substring between two strings is computed in a recursive way until no common substring can be identified. Whenever a substring is found, it is removed and the process continues by searching again for the next biggest substring. The sum of the lengths of these substrings is then scaled with the length of the strings by obtaining the commonality between the original strings. Formally, as defined in [121]:

$$comm(s_1, s_2) = \frac{2 \times \sum_i length(maxComSubString_i)}{length(s1) + length(s2)}$$

where $length$ is a function which computes the number of characters belonging to a string and $maxComSubString_i$ is the longest common substring between the strings s_1 and s_2 computed at iteration i. As for the difference function, this is based on the length of the unmatched strings that have resulted from the initial matching step. Formally, as defined in [121]:

$$diff(s_1, s_2) = \frac{uLen_{s_1} \times uLen_{s_2}}{p + (1-p) \times (uLen_{s_1} + uLen_{s_2} - uLen_{s_1} \times uLen_{s_2})}$$

where $p \in [0, +\infty)$ is a parameter used to give a different importance to the difference component of the smoa distance (typically is used $p = 0.6$) and $uLen_{s_1}$ and $uLen_{s_2}$ represent the length of the unmatched substring from the initial strings s_1 and s_2 scaled with the string length, respectively.

In particular, in our research work, three lexical matchers are used, named *Entity Name Distance Matcher*, *Comment Distance Measure* and *Entity Text Distance Matcher*. In detail, the former computes the smoa distance between the names of ontology entities, whereas, the second one computes the smoa distance between comment texts of ontology entities. The choice of the smoa distance is tied to experiment results in [121] which show how it is the most performing distance for ontology alignment problem. As for the *Entity Text Distance Matcher*, it returns the minimum value between the distances computed separately between comments and labels of ontology entities. In particular, a distance based on the vector space model [115] is used in order to compare labels or comments of ontology entities. The vector space model is an algebraic model widely used in information retrieval. It allows to determine the distance between two ontology entities by representing an entity by a vector in a space where the dimensions are terms extracted by $rdfs : label$ or $rdfs : comment$ annotations. In detail, let consider two entities e_1 and e_2 and the corresponding annotations (label or comment) a_1 and a_2. For each entity, a set of terms $T = \{t_1, t_2, \ldots, t_n\}$ is extracted by the corresponding annotations. Then, let consider $\overrightarrow{\tau} = (t_1, t_2, \ldots, t_k)$ be the vector composed of the set of terms occurring in a_1 and a_2 with no duplicates. Finally, the vector representing the i^{th} entity (with $i = 1, 2$) is as follows:

$$V_i = \{w_{i_1}, w_{i_2}, w_{i_3}, \ldots, w_{i_h}\}$$

where each w_{i_j} represents the weight of term $t_i \in T_i$. There are several ways to compute the weight values such as boolean approach or TF-IDF scheme [114]. In our work, a frequency weighting scheme is used, i.e., a weight value represents the frequency of the corresponding term in the relative annotation text. Once the annotation texts a_1 and a_2 are represented by two vectors V_1 and V_2, it is possible to compute distance between them by using the cosine of the angle between them. Let consider ϕ the angle between V_1 and V_2, the distance between e_1 and e_2 is as

follows:

$$d(e_1, e_2) = 1 - cosine(\phi).$$

In general, a **linguistic measure** computes the similarity between ontology entities by considering linguistic relations such as synonymy, hypernymy and so on. In order to compute the linguistic similarity or inversely the linguistic distance, a dictionary is needed such as WordNet[1]. Typically, the linguistic distance is computed by considering the names of entities, but also entity labels can be used.

A particular similarity measure used in this research work is referred as *Word Net Synonymy Name Distance Measure*. It uses WordNet to compute a synonymy-based distance by considering the names of entities. In detail, this distance returns the value 1 if the original strings are synonymous, otherwise it computes the traditional substring distance between all senses of the first string with the second one and then it returns the smallest of these distances. Formally, the traditional substring distance *subString* between the two strings s_1 and s_2 can be defined as follows:

$$subString(s_1, s_2) = 1 - \frac{2 \times length(maxComSubString)}{length(s_1) + length(s_2)}$$

where *length* is the function which computes the number of characters belonging to a string and $maxComSubString$ is the longest common substring between the strings s_1 and s_2.

Structural measures compute a similarity or a distance between ontological entities by considering their kinship (parents and children) within ontologies. In this work, four different structural measures, referred as *Super Hierarchy Distance Measure*, *Numbered Hierarchy Distance Measure*, *Individual Distance Measure* and *Domain and Range Restrictions Distance Measure* are considered.

In detail, the first one is based on the supersumption relation between entities in ontologies. In particular, a correspondence between two entities (class or property) inherits the confidence value related to the correspondence between their respective superentities. Formally, let e_1 and e_2 be entities of different ontologies

[1]http://wordnet.princeton.edu/

O_1 and O_2 and s_1 and s_2 be superentities, respectively, of the entities e_1 and e_2 in O_1 and O_2, and there is a correspondence $c = (s_1, s_2, \eta)$, then the distance between e_1 and e_2 computed by means of the *Super Hierarchy Distance Measure* is as follows:

$$d_{hierarchy}(e_1, e_2) = 1 - \eta.$$

The second structural measure exploited in this work is based on the number of superentities and subentities that an entity has in ontologies. Let e_1 and e_2 be entities of two different ontologies O_1 and O_2, $nsuper_1$ and $nsuper_2$ be the number of superentities characterizing, respectively, e_1 and e_2, and $nsub_1$ and $nsub_2$ be the number of subentities characterizing, respectively, e_1 and e_2, then the distance between e_1 and e_2 computed by means of the *Numbered Hierarchy Distance Measure* is as follows:

$$d_{num_{hierarchy}}(e_1, e_2) = \frac{r_{sup} + r_{sub}}{2}$$

where r_{sup} and r_{sub} represent, respectively, the distance for number of super-entities and sub-entities, calculated as follows:

$$r_{sup} = \frac{abs(nsuper_1 - nsuper_2)}{max(nsuper_1 - nsuper_2)}$$

and

$$r_{sub} = \frac{abs(nsub_1 - nsub_2)}{max(nsub_1 - nsub_2)}$$

where *abs* is a function which computes the absolute value and *max* is a function which computes the maximum value.

The third structural matcher used in this work computes an individual based distance. In detail, let e_1 and e_2 be entities of different ontologies O_1 and O_2 and ind_1 and ind_2 be the number of individuals, respectively, for entity e_1 and e_2, then the distance between e_1 and e_2 computed by means of the *Individual Distance Measure* is as follows:

$$d_{individual}(e_1, e_2) = 1 - \frac{matches}{max(ind_1, ind_2)}$$

where *matches* is the number of identical individuals between e_1 and e_2. In this measure, two individuals are considered identical if they are characterized by the same name.

Finally, the last matcher is based on the domain and range restrictions on properties in ontologies. In particular, two properties can be considered composing a correspondence, if their domain and range restrictions are similar class descriptions. Formally, let p_1 and p_2 be properties of the ontologies O_1 and O_2 to align. Let D_1 and D_2 be the sets of domain class descriptions of p_1 and p_2, respectively. Similarly, let R_1 and R_2 be the sets of range class descriptions of p_1 and p_2, respectively. Besides, let

$$C_D = \{(x_1, x_2) \in A | x_1 \in D_1 \text{ and } x_2 \in D_2\}$$

be the sets of correspondences which involve a domain class description of p_1 with a domain class description of p_2. Analogously, let

$$C_R = \{(x_1, x_2) \in A | x_1 \in R_1 \text{ and } x_2 \in R_2\}$$

be the sets of correspondences which involve a range class description of p_1 with a range class description of p_2. Starting from these sets, the domain and range class distance can be computed as follows:

$$d_{domain} = \begin{cases} \frac{\sum_{c \in C_D} f(c)}{|C_D|} & \text{if } |C_D| \neq 0 \\ 1 & \text{else} \end{cases}$$

$$d_{range} = \begin{cases} \frac{\sum_{c \in C_R} f(c)}{|C_R|} & \text{if } |C_R| \neq 0 \\ 1 & \text{else} \end{cases}$$

. Finally, the *Domain and Range Restrictions Distance Measure* computes the distance between the two properties p_1 and p_2 as follows:

$$d_{DomRan}(p_1, p_2) = \frac{d_{domain} + d_{range}}{2}.$$

2.3.1.3 Aggregation strategies

In order to combine all considered similarity measures, an aggregation strategy is needed. In general, an aggregation function is defined as:

$$\phi : \overrightarrow{s}(c) \times \overrightarrow{w} \rightarrow (0,1) \text{ with} \sum_{i=1}^{n} w_i = 1$$

where n is the number of considered similarity measures. There are several aggregation strategies, in this work three of these ones are taken in account:

- *Minimum aggregation*: this kind of strategy choices the minimum between the computed distances (for this reason this strategy leaves out the vector of weights). Formally:

$$\phi(\overrightarrow{s}(c), \overrightarrow{w}) = min\{s_1(c), s_2(c), \ldots, s_n(c)\}.$$

- *Weighted average aggregation*: this strategy computes the standard weighted average of all computed distances. Formally:

$$\phi(\overrightarrow{s}(c), \overrightarrow{w}) = \sum_{i=1}^{n} w_i h_i(c).$$

- *Ordered weighted average (OWA) aggregation* [143]: this strategy is similar to the weighted average aggregation, but applies a constant vector of weights to a reordering of the computed distances. Formally:

$$\phi(\overrightarrow{s}(c), \overrightarrow{w}) = \sum_{i=1}^{n} w_i h_{k_i}(c)$$

  where for $i \in \{1, 2, \ldots, n\}$, k_i is a reordering, such that $h_{k_i}(c) < h_{k_j}(c)$ for $i < j$.

2.3.2 Evaluating ontology alignment quality

Over years, a lot of ontology alignment systems have been developed in order to implement an ontology alignment process in an automatic way. However,

a crucial step in their adoption in real world applications is represented by the ability of determining their performances on realistic ontologies in terms of quality of produced alignments and not only. This has led to numerous researchers for designing techniques aimed at executing a systematic evaluation of these systems. Precisely, the aim of these evaluation techniques is twofold: helping users to estimate the suitability of the ontology alignment systems to their needs and helping developers of such systems to improve them. The evaluation should be held over time in order to asses, apart from absolute results, i.e., what are the properties achieved by a system, relative results, i.e., how these results compare to the results of other systems [41]. In this scenario, a coordinated international initiative, named Ontology Alignment Evaluation Initiative (OAEI), was set up in 2005 for forging a consensus on evaluation techniques to be used to asses ontology alignment systems. The evaluation strategy used by OAEI is benchmarking as described by Castro et al. [22] and summarized below. A benchmark is a test that measures the performance of a system or subsystem on a well defined task or set of tasks[1]. Evaluation should enable the measure of the degree of achievement of proposed tasks on a scale common to all methods. Benchmarks should be reproducible and non ambiguous, so that they can be used repeatedly for: (i) testing the improvement of a system with certainty and (ii) situating a system among others. The OAEI task is set up a collection of reference sets of tests, or benchmark suites, for assessing the strengths and weaknesses of the existing ontology alignment systems and to compare their evolution with regard to these references. Building benchmark suites and opportune data sets is valuable not just for groups of people who participate in the annual campaigns but for all the community, since system designers can make use of them at any time and compare their results with those of the other systems [41]. In section 2.3.2.1, a comprehensive description of the data sets provided by OAEI and exploited in this research work is given.

Another crucial topic is how to measure the evaluation results returned by the benchmarking. In literature, there is a wide range of different possible measures for evaluating ontology alignment systems including both qualitative and quantitative methods. In section 2.3.2.2, there is a comprehensive description

[1]Sill, D.: comp.benchmarks frequently asked questions version 1.0 (1996)

of the evaluation measures used by OAEI. Finally, section 2.3.2.3 presents the alignment format defined by OAEI in order to allow ontology alignment systems to produce "standardized" results aimed at easing the fair comparison.

2.3.2.1 Datasets

Good datasets are a prerequisite for a good evaluation [41]. They should allow to meet the following desiderata: the coverage of relevant aspects and the fairness of the evaluation. In the case of ontology alignment process, a dataset typically consists of at least two ontologies and a reference alignment between them. In the following, we call the combination of exactly two ontologies and, if present, a reference alignment between these ontologies a *test case*. A dataset is composed of several test cases. In [54], the authors proposed the following criteria for designing or selecting datasets for ontology matching evaluation:

- *Complexity*, i.e., how much the dataset is hard for state of the art matching systems;
- *Discrimination ability*, i.e., how much the dataset can discriminate sufficiently among various matching approaches;
- *Incrementality*, i.e., if the dataset allows for incrementally discovering the weaknesses of the tested systems;
- *Monotonicity*, i.e., the matching quality measures calculated on subsets of gradually increasing size converge to the values obtained on the whole dataset;
- *Correctness*, i.e., a reference alignment is available for the dataset, which allows to divide generated correspondences into correct and incorrect ones.

From 2005, OAEI is engaging in developing of several datasets in order to cover as much as possible the desired criteria discussed above. The first developed dataset is named *benchmark*. The benchmark dataset deals with the topic of scientific publications. It consists of a large set of artificial test cases which alter an initial ontology in order to match it to one of its variants. Modifications

concern both the element labels, e.g., replacing them by random labels, and the structure, e.g., deleting or inserting classes in the hierarchy. In addition, the dataset comprises four other real ontologies that have to be matched to the reference ontology. The ontologies are described in OWL-DL and serialized in the RDF/XML format. Moreover, each benchmark test case provides also a reference alignment useful for the evaluation task. Table 2.1 shows the general organization of the benchmark dataset.

Table 2.1: Benchmark track description

Test case Range	Description
# 101-104	The ontologies under alignment are the same or the first one is the "OWL Lite restriction" of the second one.
# 201-210	The ontologies under alignment have the same structure, but different lexical and linguistic features.
# 221-247	The ontologies under alignment have the same lexical and linguistic features, but different structure.
# 248-266	The ontologies under alignment have different lexical, linguistic and structure features.
# 301-304	The ontologies under alignment are real world cases.

The benchmark dataset will be used in all experiments performed in our research work. It has been chosen for its completeness: it considers both artificial and real test cases.

2.3.2.2 Evaluation measures

In order to evaluate alignment quality, OAEI considers different evaluation measures. In particular, in our research work, we consider the following ones:

- *compliance measures* which evaluate the degree of conformance of the alignment ontology systems to what is expected;
- *performance measures* which measure non functional but important features of the algorithms (such as speed).

In detail, compliance measures are used for computing the quality of the output provided by a system compared to a reference output. It is worth noting that

this reference output is not always available and not always consensual. However, for the purpose of benchmarking, we can assume that it is desirable to provide such a reference [39]. The most commonly used and understood compliance measures are: *precision*, *recall* and *F-measure*. They derive from the information retrieval field [113] and compute the quality of an ontology alignment by comparing it with a reference alignment R, as depicted below:

Definition 5 (Precision) *Given a reference alignment R, the precision of an alignment A is given by*

$$Pr(A, R) = \frac{|R \cap A|}{|A|}.$$

Definition 6 (Recall) *Given a reference alignment R, the recall of an alignment A is given by*

$$Rec(A, R) = \frac{|R \cap A|}{|R|}.$$

Definition 7 (F-measure) *Given a reference alignment R and a number α between 0 and 1, the F-measure of an alignment A is given by*

$$M_\alpha = \frac{Pr(A, R) \cdot Rec(A, R)}{(1 - \alpha) \cdot Pr(A, R) + \alpha \cdot Rec(A, R)}.$$

If $\alpha = 1$, then the F-measure is equal to precision and if $\alpha = 0$, the F-measure is equal to recall. In between, the higher α, the more importance is given to precision with regard to recall. Very often, the value $\alpha = 0.5$ is used, i.e.,

$$M = \frac{2 \times Pr(A, R) \times Rec(A, R)}{Pr(A, R) + Rec(A, R)},$$

the harmonic mean of precision and recall.

Instead, performance measures (or non-functional measures) asses the resource consumption necessary for aligning two ontologies. According [39], they can be used when the ontology alignment systems are 100% compliant or balanced against compliance. Unlike the compliance measures, performance measures depend on the benchmark processing environment and the underlying ontology management system [39] and, as a consequence, it is rather difficult to obtain objective evaluations. The most common performance measures are:

- *Speed*: it is measured in amount of time taken by the algorithms for accomplishing their alignment tasks. If user interaction is required, one has to ensure to effectively measure the processing time of the machine only.

- *Memory*: the amount of memory used for accomplishing the alignment task marks another performance measure. Due to the dependency with underlying systems, it could also make sense to measure only the extra memory required in addition to that of the ontology management system (but it still remain highly dependent).

2.3.2.3 Alignment format

In section 2.3.1, we have given a formal definition of an alignment which is rather abstract since it does not provide a concrete format that can be used for expressing these alignments. "Reifying" alignments in a standardised format can be very useful for several reasons, for example, for collecting hand-made or automatically created alignments in libraries that can be used for linking two particular ontologies or for comparing the results with each other ontology alignment systems or with possible "standard" results [42]. The desired requirements that an alignment format should satisfy are [42]:

- being Web ready: in particular using URIs and semantic web languages (XML, RDF);

- being ontology language independent: this allows alignments between ontologies written in different languages;

- being simple so that current ontology matching tools can manipulate it without having to implement a full-fledged knowledge representation language;

- being expressive so that it can cover an important part of the usable relations between ontologies;

- supporting many different uses, i.e., it should not be committed to one particular usage.

All these features are satisfied by the alignment format proposed in [37] and used in OAEI campaigns. In detail, this alignment format characterizes an alignment description with the following components:

- a *set of correspondences* which express the relation holding between entities of the first ontology and entities of the second ontology;
- an *arity* noted with, 1 for injective and total, ? for injective, + for total and * for none.

The alignment format is expressed in RDF/XML so that it is easy to parse, but, this requires that entities are identifiable by a URI. A full example of alignment format in RDF is reported in listing 2.2. In detail, it describes a many-to-many alignment between two bibliographic ontologies. In particular, it contains two correspondences that associate *reviewedarticle* with article and journalarticle with *journalarticle* respectively. These correspondences are characterized by an equivalence relation and a confidence measure equal to 0.64 in the former case and 1.0 in the latter.

2.3.3 Use cases

Ontology alignment process is considered to be a crucial preliminary step for a lot of real-life applications. In this section, we present some of these applications as described in [38] highlighting the need for and use of ontology alignment process.

2.3.3.1 Web service integration

Web service discovery is the process of finding web services that meet a given requester goal. Both the requester goal and the service capability, i.e., requested functionality and provided functionality, are defined declaratively and in a machine-processable way using one or more domain-specific ontologies. In this scenario, two different problems can arise:

1. The descriptions of the capability or the goal use several domain ontologies that are characterized by conflicts due to either the definition of a different conceptual model or the use of a different ontology language.

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<!DOCTYPE rdf:RDF SYSTEM "align.dtd">
<rdf:RDF
xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/
                                                    alignment'
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
 <Alignment>
  <xml>yes</xml>
  <type>**</type>
  <onto1>http://www.example.org/ontology1</onto1>
  <onto2>http://www.example.org/ontology2</onto2>
  <map>
   <Cell>
    <entity1 rdf:resource=
     'http://www.example.org/ontology1#reviewedarticle'/>
    <entity2 rdf:resource=
     'http://www.example.org/ontology2#article'/>
    <measure rdf:datatype='&xsd;float'>0.64</measure>
    <relation>=</relation>
   </Cell>
  </map>
  <map>
   <Cell>
    <entity1 rdf:resource=
     'http://www.example.org/ontology1#journalarticle'/>
    <entity2 rdf:resource=
     'http://www.example.org/ontology2#journalarticle'/>
    <measure rdf:datatype='&xsd;float'>1.0</measure>
    <relation>=</relation>
   </Cell>
  </map>
 </Alignment>
</rdf:RDF>
```

Listing 2.2: A full example of alignment format

2. The capability and the goal are expressed using ontologies that describe a common domain, but are different between them. Obviously, the discovery process has to find suitable services despite the use of different terminologies for the goal and the capability.

These heterogeneity problems must be solved in order to enable the reuse of (possibly conflicting) ontologies for goal and capability descriptions, and, an ontology alignment process is a suitable method for accomplishing this task.

2.3.3.2 Catalog matching

Many e-Commerce applications are based on the publication of electronic catalogs which present the goods on sale and allow customers to select the goods they need. However, a very important obstacle to the success of distributed e-Commerce applications is the problem of interoperating different catalogs. Indeed, many systems require participant parties to perform very costly operations on their local catalogs to enter into the system, and this is a tremendous barrier for newcomers. A typical instance of this situation are e-Marketplaces, i.e., electronic malls where different sellers provide their goods in a common environment. The problem of this application is that each provider typically owns a local catalog, in which goods are organized according to criteria that suit its internal business processes. However, in order to take part in the marketplace, providers should translate their catalogs into a common catalog, which will be presented to users as a single access point to what is sold in the marketplace. Notice that, in principle, this translation is required for each marketplace in which a company is involved, which means that a potentially very high number of catalogs should be maintained at the same time by each company. This is considered one of the strong barriers against the success of e-Marketplaces. This scenario would be much more appealing if we could provide means for aligning catalogs so as to strongly reduce efforts for newcomers.

2.3.3.3 P2P information sharing

In last years, Peer-to-Peer (P2P) information sharing systems have had a variety of implementations and are widely used on the Web. They use a simple or more complex schema such as databases or ontologies in order to describe contents to be exchanged. Currently, most of the systems are based on the use mappings between peer schema a priori. However, in P2P settings, assumptions that all parties agree on the same scheme, or that all parties rely on one global scheme

(as in data integration) are not possible. Peers come and go, import multiple schema into the system, and have a need to interoperate with other nodes at run-time. In this activity, a scheme alignment represents the main process to enable a full nodes' interoperation. Namely, when two peers "meet" on the network, they establish mappings between their schema in a (semi) automatic alignment discovery process. Automation of the schema alignment discovery process will create a great advance for the P2P information sharing systems on the Semantic Web. Peers will be able to bring to the system various schema, "align" them to the schema of other peers on the network with no (or minimal) user involvement, and exchange requests and data in a decentralized, collaborative manner.

2.3.3.4 Semantic query processing

Semantic query processing is a run-time scenario where a user specifies the output of a query (e.g., the SELECT clause in SQL), and the system figures out how to produce that output (e.g., by determining the FROM and WHERE clauses in SQL). The user's specification is stated in terms of concepts familiar to her, which may not be the same as the names of elements specified in the database schema. Therefore, in the first phase of processing the query, the system must map the user-specified concepts in the query output to schema elements. This is a natural application of the ontology alignment process.

2.3.4 An ontology alignment example

For better understanding an ontology alignment process, let us consider an example by using the ontologies depicted in Figs. 2.2 and 2.3, respectively, referred as O_1 and O_2. As aforementioned, these two ontologies are related to the same domain. A reasonable alignment between the two ontologies is given in Table 2.2. Each line contains the two corresponding entities, each one belongs to one of the considered ontologies. In Fig. 2.5, the provided alignment is shown graphically. Whereas the alignment might seem obvious to some readers, others might disagree [36]. The common agreement on alignments is not easy. Hence, the birth of an organization, the OAEI, aimed to establish common evaluation methods (see section 2.3.2). In order to complete our example, listing 2.3 presents a frag-

ment of the considered alignment in the the RDF-based representation proposed by OAEI (see section 2.3.2.3). The first lines are dedicated to ontology names and their URIs. In the following, the description of some cells of the considered alignment including the kind of relation and the confidence value is given.

Table 2.2: An example of alignment

Ontology 1	**Ontology 2**
Object	Thing
Vehicle	Conveyance
Car	Car
Speed	Speed
HasSpeed	HasProperty
Fiat 500	Paul's Fiat 500
160 km/h	Fast

2.4 State of the art about the Ontology Alignment

Over years, a lot of strategies have been investigated to perform an ontology alignment process. Good surveys are provided in [78][111][134]. However, in this research work, we take in account a new classification of ontology matching techniques described in [118] and based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. A detailed discussion about this classification is given in subsection 2.4.1, whereas, a description of more recent ontology alignment systems is given in subsection 2.4.2.

2.4.1 Classification

Initially, Shvaiko et al. [118], taking inspiration from [111], distinguish between *elementary* (individual for Rahm et al. [111]) and *combination* techniques. In

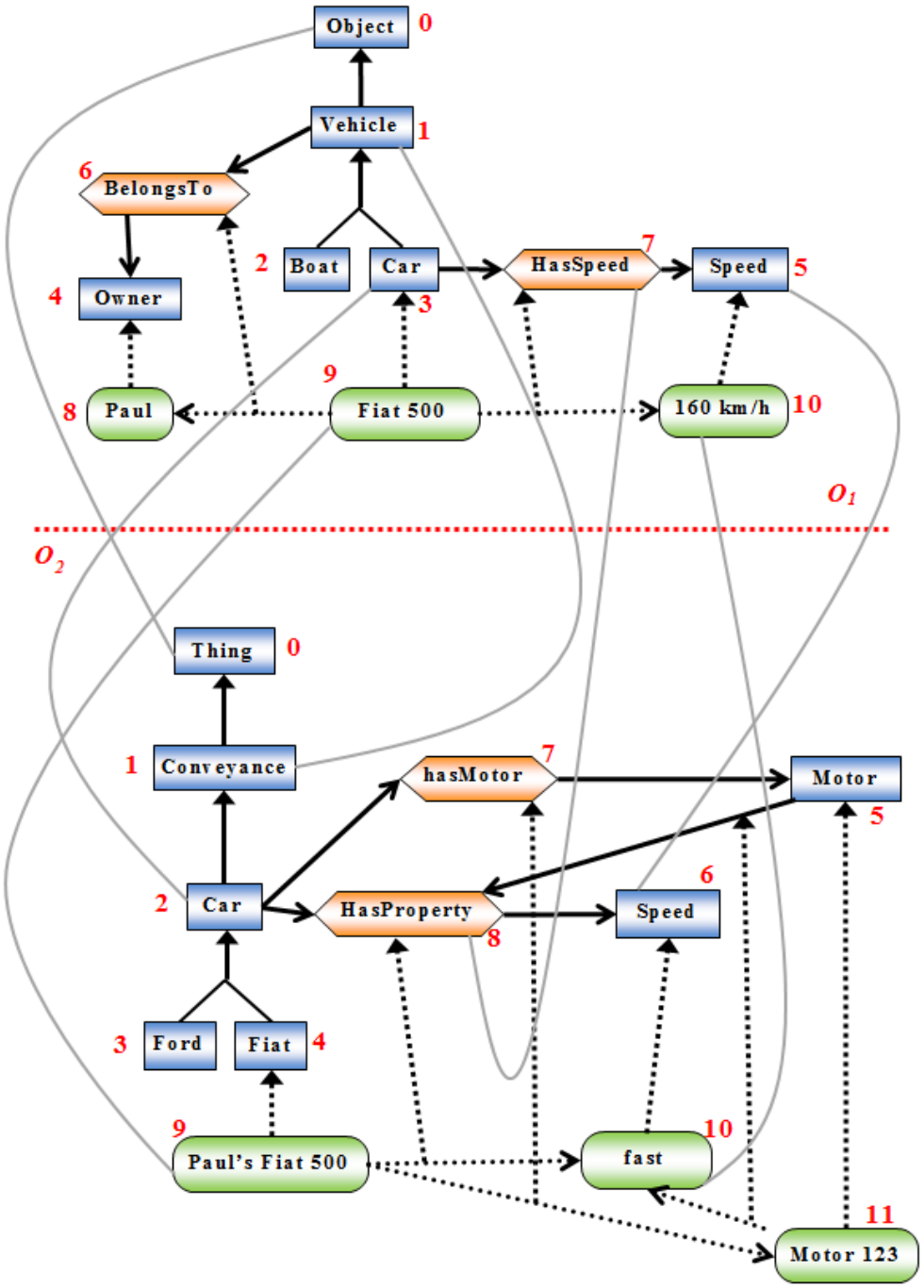


Figure 2.5: An example of ontology alignment

particular, the former computes alignments based on a single matching criterion, whereas, the latter performs a combination of individual matchers. Therefore, the

```
<rdf:RDF>
  <Alignment>
    <xml>yes</xml>
    <level>0</level>
    <type>11</type>
    <onto1>ontology1.owl</onto1>
    <onto2>ontology2.owl</onto2>
    <uri1>http://aifb.uni-karlsruhe.de/ontology1.owl</uri1>
    <uri2>http://aifb.uni-karlsruhe.de/ontology2.owl</uri2>
    <map>
      <Cell>
        <entity1 rdf:resource='ontology1.owl#Object'/>
        <entity2 rdf:resource='ontology2.owl#Thing'/>
        <measure rdf:datatype='XMLSchema#float'>1</measure>
        <relation>=</relation>
      </Cell>
      <Cell>
        <entity1 rdf:resource='ontology1.owl#Vehicle'/>
        <entity2 rdf:resource='ontology2.owl#Conveyance'/>
        <measure rdf:datatype='XMLSchema#float'>1</measure>
        <relation>=</relation>
      </Cell>
      ................
    </map>
  </Alignment>
</rdf:RDF>
```

Listing 2.3: A fragment in RDF-representation of alignment presented in Fig. 2.5

classification focuses on individual matchings. In turn, elementary matchings can be divided in *instance-based* and *schema-based*: in order to perform a matching process, the former uses all entities of an ontology including the instances, the latter uses only entities as classes and properties. For classifying elementary schema-based matching techniques, Shvaiko et al. [118] introduce two synthetic classifications (see Fig. 2.6), based on the most salient properties of the matching dimensions. These two classifications are:

- *Granularity/Input Interpretation classification* is based on (i) granularity of match, i.e., element- or structure-level, and then (ii) on how the techniques generally interpret the input information;

- *Kind of Input classification* is based on the kind of input which is used by elementary matching techniques.

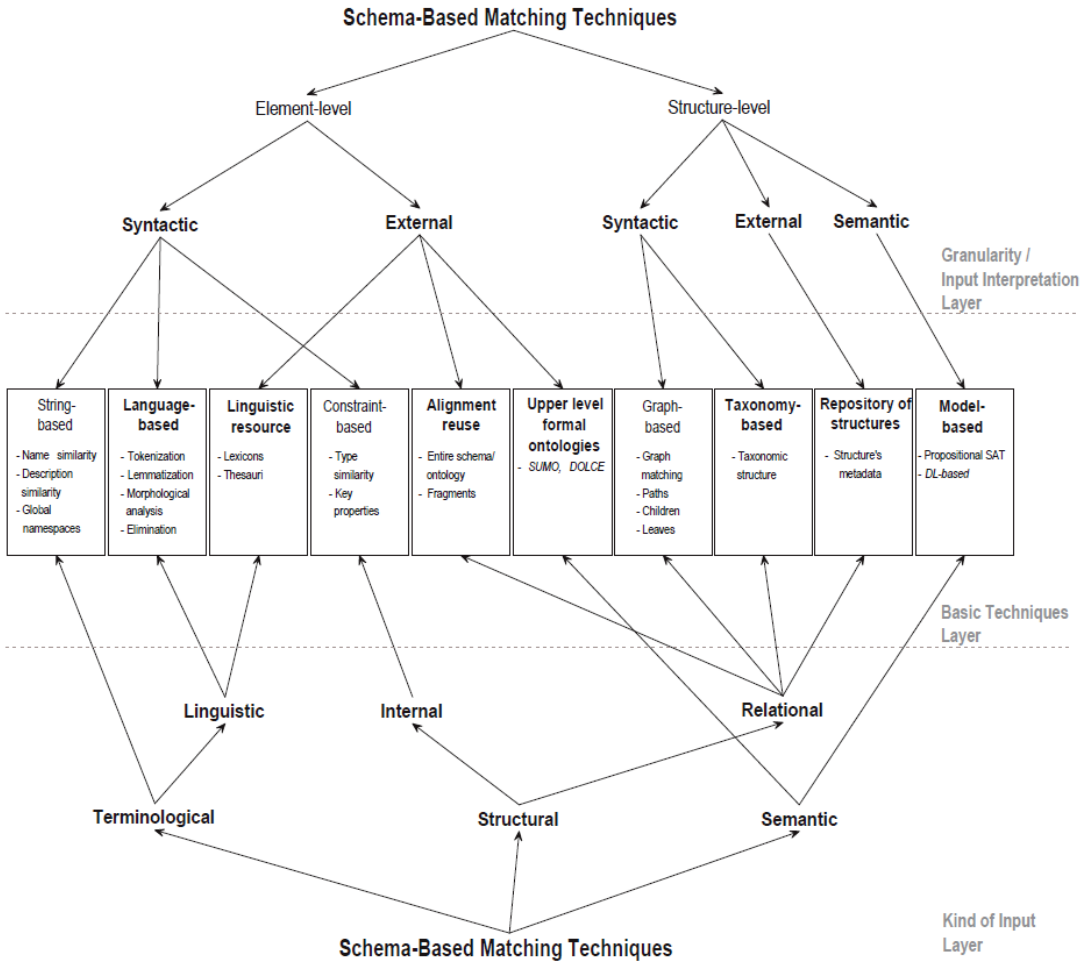


Figure 2.6: A classification of elementary schema-based matching approaches [118]

The classification depicted in Fig. 2.6 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kind of manipulated objects) way in order to reach the leaves representing the *basic techniques layer*. In detail, elementary ontology alignment systems are classified by the *Granularity/Input interpretation layer* according to the following criteria:

- *Element-level vs structure-level*: element-level ontology alignment systems compute mapping elements by analyzing entities in isolation, ignoring their relations with other entities, whereas, structure-level techniques compute mapping elements by analyzing how entities appear together in a structure.

- *Syntactic vs external vs semantic*: syntactic techniques interpret the input only in function of its structure following some clearly stated algorithms; external are techniques exploiting auxiliary (external) resources of a domain such as dictionaries and common knowledge in order to interpret the input; semantic techniques use some formal semantics (e.g., model-theoretic semantics) to interpret the input and justify their results.

Distinctions between classes of elementary matching techniques in the *Basic Techniques layer* are motivated by the way a matching technique interprets the input information in each concrete case. For example, a label can be interpreted as a string (a sequence of letters from an alphabet) or as a word or a phrase in some natural language, a hierarchy can be considered as a graph (a set of nodes related by edges) or a taxonomy (a set of concepts having a set-theoretic interpretation organized by a relation which preserves inclusion) [118].

The *Kind of Input layer* classification deals with the type of input considered by a particular technique. In detail, it is divided into two levels [118]:

- the first level is categorized depending on which kind of data the algorithms work on: strings (terminological), structure (structural) or models (semantics). The two first ones are found in the ontology descriptions, the last one requires some semantic interpretation of the ontology and usually uses some semantically compliant reasoner to deduce the correspondences;

- the second level of this classification decomposes further these categories if necessary: terminological methods can be string-based (considering the terms as sequences of characters) or based on the interpretation of these terms as linguistic objects (linguistic). The structural methods category is split into two types of methods: those which consider the internal structure of entities (e.g., attributes and their types) and those which consider the relation of entities with other entities (relational).

Finally, in Fig. 2.6, techniques which are marked in italic (techniques based on upper level ontologies and DL-based techniques) have not been implemented in any ontology alignment system yet. However, Shvaiko et al. [118] introduce them because their appearance seems reasonable in the near future.

2.4.2 Existing Ontology Alignment Systems

Over the years, the relevant importance of ontology alignment problem and its complexity have led to need of developing automatic and semi-automatic ontology alignment systems based on several strategies. However, only recently, some computational intelligence techniques such as machine learning and evolutionary algorithms have been investigated for solving the ontology alignment problem. Hereafter, we give a brief overview of the most known ontology alignment systems dividing them in two groups according to whether they are based on techniques belonging to computational intelligence or not.

2.4.2.1 Deterministic Ontology Alignment Systems

One of the first ontology alignment systems has been PROMPT [103], i.e., a semi-automatic system which does not produce in output an alignment, but rather a set of suggestions useful to users for matching classes and properties and, as a consequence, building a mapping file. Among the first automatic ontology alignment systems, instead, there is Cupid [87]. It computes a mapping between two ontologies by choosing pairs of entities with an aggregated similarity value higher than a specific threshold. The aggregated similarity value is computed by performing a weighted average of two coefficients obtained, respectively, by computing a linguistic and structural-based matching. The weights and the threshold are manually set. Since then, several automatic ontology alignment systems have been implemented by taking into account different techniques. Among the most recent ones, it is possible to mention COMA++[33], CODI [101], Ef2Match [137], RiMOM [136], ASMOV [76], CIDER [57] and so on. In particular, COMA++ performs the match operation by iteratively executing three steps: *element identification* to determine the relevant schema elements for matching, *matcher execution* applying multiple matchers to compute the element similarities, and *simi-*

larity combination to combine matcher-specific similarities and derive a mapping with the best correspondences between the elements. The aggregation strategies exploited by COMA++ are Max, Min, Average, and Weighted whose the weighting scheme is manually set. RiMOM is a tool for ontology alignment that combines different strategies and aims to find the optimal alignment results: edit-distance based strategy, statistical learning based strategy, and three similarity-propagation based strategies. Each strategy is based on one kind of ontological-information/approach. Depending on their label and structure similarity factors, the algorithm will favor one or the other kind of strategy, and for this purpose, it uses heuristic rules. Finally, ASMOV automates the ontology alignment process by performing two steps: the computation of a pre-alignment starting from a similarity matrix and a semantic verification process to ensure which the pre-alignment does not contain semantic inconsistencies. In detail, the similarity matrix is built by combining different similarity measures based on four ontology features (lexical elements such as labels and comments, relation structure, internal structure such as domain and ranges of properties, and extension) through a weighted average. The involved weights are chosen experimentally by using a dataset. For instance, for the OAEI competition 2010, ASMOV optimizes the weights by exploiting the same dataset provided for the OAEI competition 2008.

2.4.2.2 Computational Intelligence based Ontology Alignment Systems

In last years, some works have explored computational intelligence techniques for addressing the ontology alignment problem. In particular, the designed systems follow two general approaches: 1) facing directly the ontology alignment problem; 2) producing alignments by addressing the nested meta-matching problem.

Among the first class of systems, we can include GLUE [34], i.e., an ontology alignment system that exploits machine learning to compute semantic mappings between concepts stored in different ontologies. In detail, by considering two distinct ontologies, the mapping discovery process between their concepts is based on the measure of similarity which is defined through the joint probability dis-

tribution computed by means of two base learning techniques applied on the ontology instances. Different from GLUE, in [135], a genetic algorithm-based ontology matching process, named GAOM, is proposed. In detail, in their work, the authors model the problem of ontology matching as a maximum optimization problem of a mapping between two compared ontologies. Each ontology is characterized by an associated set of extensional and intentional features. GAOM performs a matching process through a genetic algorithm whose the fitness function is defined as a global similarity measure function based on ontology feature sets. In addition, in [109], genetic algorithms are exploited in order to introduce an extraction method for searching an optimal or near optimal mapping between two ontologies. The proposed genetic algorithm employs a structured based weighting model, named "coincidence based model", as its fitness function to score different possible mappings. Finally, MapPSO [16], instead, addresses the ontology alignment problem as a minimum optimization problem to be resolved through the discrete particle swarm optimization. In detail, MapPSO exploits a fitness function depending on the similarity values computed by performing a combination of lexical, linguistic and structural matchers. MapPSO employs aggregation techniques (minimum, weighted average aggregation, ordered weighted average aggregation) whose weights are manually set.

Several works also belong to the second class of systems that face how to efficiently set ontology alignment process parameters. One of the first works which deals with the similarity aggregation problem by exploiting evolutionary algorithms is GOAL (Genetics for Ontology Alignments) [90]. This system computes, by means of a genetic algorithm, the optimal weight configuration for a weighted average aggregation of several similarity measures by considering a reference alignment. Indeed, GOAL uses a method of evaluation based on one of these conformance measures: precision, recall and F-measure. The same idea of implementing a meta-matcher to combine multiple similarity measures through genetic algorithms is developed in a more recent work [98]. In addition, also in [53], the authors try to optimize the combination of similarity measures by means of a genetic algorithm but, different from the previous works, they focus on optimizing all ontology alignment parameters, including the threshold value in the chromosome. However, these meta-matching methods have a relevant drawback which

affects strongly their applicability: they require an *a priori* knowledge about ontologies under alignment in order to select the most suitable set of the weights. In fact, they align ontologies by using the optimal combination of weights obtained for a pair of ontologies with the same features whose a reference alignment is known.

2.5 Ontology Alignment: Open issues

Despite many component matching solutions have been developed so far, there is no integrated solution that is a clear success, which is robust enough to be the basis for future development, and which is usable by non expert users [117]. This section aims at analyzing the key trends and open issues of the ontology matching field, by highlighting the contribution of this research work for facing some of them. In detail, the open issues discussed in [117] are:

- *large scale evaluation*: the fast growth of different matching approaches makes the issues of their evaluation and comparison more stringent. There are many points to be faced in ontology matching evaluation in order to empirically prove the matching technology to be mature and reliable. In particular, there is a need for more accurate evaluation quality measures (initial steps towards these have been presented in section 2.3.2.2);

- *quality*: in spite of several ontology matching systems have been developed, none of them seems capable of producing high quality alignments on different domains and different problem instances. Therefore, designing generic ontology alignment systems which are characterized by acceptable results is a research challenge [133];

- *performance of ontology-matching techniques*: although performance is of prime importance in many dynamic applications, for example, where a user can not wait too long for the system to respond, it is not yet a trend in this field. Indeed, the results of the anatomy track of OAEI-2011 [40], where only two systems take a time under 10 minutes, is a clear aspect of this poor consideration of performance. However, new application scenarios of the

ontology alignment process such as semantic search and Web service composition stress the importance of computational complexity considerations, even if it is always strong the opinion that it is worth doing computational optimizations only once the underlying basic techniques are stable.

- *discovering missing background knowledge*: one of the sources of difficulty for the alignment tasks is that ontologies are designed with certain background knowledge and in a certain context, which unfortunately do not become part of the ontology specification, and, thus, are not available to matchers. Hence, the lack of background knowledge increases the difficulty of the alignment task, e.g., by generating a lot of ambiguities. Several strategies have been explored in order to address the problem of the lack of background knowledge such as declaring the missing axioms manually as a pre-match effort or reusing previous match results, but, without a clear success.

- *uncertainty in ontology matching*: because the syntactic representation of the ontologies cannot completely describe the semantics of different ontologies, automatic matching techniques bring with them a degree of uncertainty [93] related to correctness of produced alignments. The issue of handling uncertainty in ontology alignment scenario has been faced with several strategies such as by using the notion of probabilistic schema mappings. However, in spite of the work done, there is still a need to understand better the foundations of modeling uncertainty in ontology alignment scenario in order to enhance detection of matchings causing inconsistencies.

- *matcher selection and self-configuration*: there are a lot of matchers that are available so far. However, often these systems perform well in some cases and not so well in some other cases. This leads to the following issues: 1) matcher selection, i.e, which similarity measures should be used for a particular instance of ontology alignment problem; 2) matcher combination, i.e., how selected similarity measures should be combined; and 3) matcher tuning, i.e., how to tune and adapt automatically matching solutions to the settings in which an application operates. So far, only

few ontology alignment systems face the ontology meta-matching problem [98][53], but performing or semi-automatic process or procedure requiring *a priori* knowledge.

- *user involvement*: another open issue is whether it is possible to produce accurate alignments without any user intervention or human knowledge? However, if an ontology alignment system considers user involvement, thus it must provide a way for users to analyze the results of an ontology matching process and understand the characteristics of the source ontologies. In other word, there is a need for new human-readable ontology-matching interfaces. So far, there have only been few studies focused on the ergonomic aspect of elaborating alignments, either for designing them manually or for checking and correcting them.

In our research work, we focus on second and third issues by attempting to design an efficient ontology alignment system both in terms of alignment quality and computational effort. In order to achieve this result, we investigate the application of a new class of approximate methods, named memetic algorithms, for facing the ontology alignment problem following two directions: solving directly the ontology alignment problem as an optimization one and producing alignments by addressing of nested meta-matching problem. The research is resulted in two systems, respectively, named *MemeOptiMap* and *MemeMetaMap*, which, after some experiments on a well-known dataset, have shown competitive performances. However, in particular, *MemeMetaMap* allows to address also other presented open issues such as the self-configuration, and, as side effect, it deals with the issues related to background knowledge and user involvement by removing them upstream. Indeed, *MemeMetaMap* efficiently works regardless of *a priori knowledge* about ontologies and user intervention.

Chapter 3

An Emergent Search Paradigm: The Memetic Algorithms

In this chapter, we introduce the so-called Memetic Algorithms (MAs)(see section 3.1) which represent the main methodology used in our research work to address the ontology alignment problem. MAs are meta-heuristic search methods which combine genetic algorithms with local search strategies. Therefore, in order to allow readers to understand MAs' general structure (see section 3.5), basic concepts about search algorithms (see section 3.2), some local search methods (see section 3.3) and genetic algorithms (see section 3.4) will be described. Then, in section 3.6, we present an emergent extension of conventional MAs aimed at overcoming some their limitations. The chapter ends with a discussion about the techniques used to execute performance comparisons among MAs, and, in general, among different approaches (see section 3.7).

3.1 Introduction

Back in the late 80s, the term "Memetic Algorithms" (MAs) was given birth to denote a wide family of metaheuristics that attempted to merge concepts from clearly separated methodologies such as Evolutionary Algorithms (EAs) and Local Search (LS) methods. This marriage makes MAs intrinsically capable of exploiting *all available knowledge* about the problem under study [96]. The

philosophy of incorporating problem domain knowledge is fully represented by the term "memetic" coined by Dawkins in [29] for denoting an analogous to the gene in the context of cultural evolution [95]. Quoting Dawkins:

> *Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation.*

This characterization of a meme inspires that information exchanged in a cultural evolution process is not simply transmitted unaltered between individuals, but it is processed and enhanced by the communicating parts. This enhancement is accomplished in MAs by incorporating heuristics, approximation algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. [96]. Initially, MAs had to suffer, but they are now becoming increasingly popular, as demonstrated by several works applying them (see section 3.5.1). In the next sections, we provide a in-depth description about MAs (see section 3.5), focusing on their technical and formal features starting with basic concepts related to local search and population-based search (see section

3.2 Local vs Population-based algorithms

An *algorithm* is a detailed step-by-step procedure for solving a *computational problem*. As described in [96], a computational problem P denotes a class of algorithmically-doable tasks characterized by an input domain set of *instances* denoted I_P. For each instance $x \in I_P$, it is possible to denote with $sol_P(x)$ the set of *feasible* solutions for problem P given instance x. The research is expected to find an algorithm that solves problem P. This means that the designed algorithm, given instance $x \in I_P$, must return at least one element y from a set of answers $ans_P(x)$ (also called *given solutions*) that satisfies the requirements of the problem. However, depending on the kind of answers expected, computational problems can be classified into different categories [96]:

- finding *all* solutions in, i.e., *enumeration* problems.
- counting *how many* solutions exist in, i.e., *counting* problems.
- determining whether the set is *empty or not*, i.e., *decision* problems.
- finding a solution in maximizing or minimizing a given function, i.e., *optimization* problems.

In our research work, we focus on the last possibility, that is, a problem instance will be considered *solved* by finding a certain feasible solution, i.e. either finding an optimal $y \in sol_P(x)$ or giving an indication that no such feasible solution exists. An algorithm is said to solve problem P if it can fulfill this condition for any given instance $x \in I_P$. However, this definition is too wide, therefore, we give a more restrictive characterization for our problems of interest resulting in the so-called *combinatorial optimization problems*. In detail, these are a special subclass of computational problems characterized by the following features for each instance $x \in I_P$ [96]:

- the cardinality of $sol_P(x)$ is finite;
- each solution $y \in sol_P(x)$ has a *goodness integer value* $m_P(y, x)$ obtained by means of an associated objective function m_P;
- a partial order $\prec_P$ is defined over the set of goodness values returned by the objective function, allowing determining which of two goodness values is preferable.

An instance $x \in I_P$ of a combinatorial optimization problem P is solved by finding the best solution $y* \in sol_P(x)$ i.e., finding a solution $y*$ such that no other solution $y \prec_P y*$ exists if $sol_P(x)$ is not empty. Typically, $\prec_P$ defines a total order. In this case, the best solution is the one that maximizes (or minimizes) the objective function.

In order to find at least one of the optimal solutions for a given instance, a *search algorithm* must be used. Before describing search algorithms, it is necessary to discuss three entities: search space, the neighborhood relation, and the guiding function. The *search space* for a combinatorial problem P is a set $S_P(x)$ whose elements are characterized by the following properties:

- each element $s \in S_P(x)$ represents at least one answer in $ans_P(x)$;
- at least one optimal element $y*$ of $sol_P(x)$ is represented by one element in $S_P(x)$.

Each element of $S_P(x)$ is called a *configuration* and it is related to an answer in $ans_P(x)$ by a *growth function* $g : S_P(x) \rightarrow ans_P(x)$. It is worth noting that the first requirement refers to $ans_P(x)$ and not to $sol_P(x)$. This implies that some configurations in the search space may correspond to infeasible solutions. The role of the search space is to provide a "ground" on while the search algorithm will act, and of course, indirectly moving in the image set $ans_P(x)$ [96]. Important properties of the search space are related with the accessibility relationships between the configurations which, in turn, are dependent of a *neighborhood function* $N_P : S_P \rightarrow 2^{S_P}$. This function assigns to each element $s \in S_P(x)$ a set $N_P(s, x) \subseteq S$ of neighboring configurations of s. The set $N_P(s, x)$ is named the *neighborhood* of s and each member $s' \in N_P(s, x)$ is named a *neighbor* of s. Typically, the elements of $N_P(s, x)$ are not explicitly listed, but it are implicitly defined by using a set of possible *moves*, which represent transitions between configurations. The last entity, the *guiding function*, associates to each configuration a value that assesses the quality of the solution. Formally, it is a function $F_g : S_P \rightarrow \mathcal{F}_P$ where $\mathcal{F}_P$ is a set whose elements are termed fitness values (typically $\mathcal{F}_P \equiv \mathbb{R}$) and it is characterized by a partial order $\prec_{\mathcal{F}_P}$ on $\mathcal{F}_P$ (typically, but not always, $\prec_{\mathcal{F}_P} \equiv <$). The behavior of the search algorithm will be "controlled" by these fitness values. It is worth noting that for optimization problems there is an obvious direct connection between the guiding function F_g and the objective function m_P [96].

The combination of a certain problem instance and the aforementioned three entities (search space, neighborhood relation, guiding function) induces a so-called *fitness landscape* [77]. Essentially, a fitness landscape can be defined as a weighted digraph, in which the vertices are configurations of the search space and the arcs connect neighboring configurations [96]. The weights represent the difference of the guiding function of the endpoint configurations. Therefore, the search can be seen as the process of "navigating" the fitness landscape using the information provided by the guiding function. This very powerful metaphor allows inter-

preting the search progress in terms of well-known topographical objects such as peaks, valleys, mesas, etc.. Associated with this definition of fitness landscape, there is the important notion of *local optimum*. In detail, a local optimum is a vertex of the fitness landscape whose guiding function value is better than the values of all its neighbors. It is worth noting that the notion of local optimum is not intrinsic to a problem instance as it is, sometimes, erroneously considered. This is related to the fact that different moves produce different neighborhoods, and, as a consequence, different fitness landscapes, even when the same problem instance is studied.

All aforementioned definitions naturally lead to the notion of *local search algorithm*. A local search algorithm starts from a configuration $s_0 \in S_P(x)$ generated at random or constructed by other algorithms. Subsequently, it iterates using at each step a transition based on the neighborhood of the current configuration until a certain termination criterion is met. The selection of the particular type of moves (also known as *mutation*) to use does certainly depend on the specific characteristics of the problem and the representation chosen.

Local search algorithms are thus characterized by keeping a single configuration at a time. The immediate generalization of this behavior is the simultaneous maintenance of k, $(k \geq 2)$, configurations. The term *population-based search algorithms* has been coined to denote search techniques behaving this way. The availability of simultaneously managing several configurations enables the use of new powerful methods for traversing the fitness landscape in addition to the standard mutation operator. The most common of these methods is known as the *recombination operator*. In essence, recombination can be described as a process in which a set of n configurations (informally referred to as *parents*) is manipulated to create a set of m new configurations, called *offspring*. The creation of these descendants implies the identification and combination of features extracted from the parents.

In the next sections, some examples of local search methods (see section 3.3) and an example of population-based search such as genetic algorithms (see section 3.4) are described in a more detailed way.

3.3 Local search methods

The local search paradigm derives its name from the kind of moves computed for producing a *neighbor configuration*. Indeed, these moves, known also as *mutations*, are usually defined as "local" modifications of some part of a configuration, where "locality" refers to the fact that the move is done on a single solution to obtain another single solution. Therefore, local search algorithms are characterized by keeping a single configuration at a time. All local search approaches are known for their efficiency, however, they suffer from the same drawback: they tend to get stuck in local optima (see section 3.2).

In this section, examples of local search methods are given. In particular, hereafter, we present three variants of the Hill Climbing Search and the Simulated Annealing.

3.3.1 Three variants of Hill Climbing

In general, the Hill Climbing Search is a greedy strategy which performs iterative search for optimum solution in the *neighborhood* of a candidate. The algorithm starts from an arbitrary candidate solution generated at random or constructed by other algorithms. At each iteration, the algorithm changes the current solution by typically applying a mutation operator in order to find a better solution. If the change improves the current candidate, then the new one becomes the current one. The algorithm continues until a certain termination criterion is met. Typical criteria are the realization of a pre-specified number of iterations, not having found any improvement in the last m iterations, or even more complex mechanisms based on estimating the probability of being at a local optimum [24].

In literature, there are several variants of Hill Climbing search depending on how the next solution is tried. In this work, three variants are considered: the *Simple Hill Climbing algorithm* (see Table 3.1) which chooses the first closer node to solution, the *Stochastic Hill Climbing search* (depicted in the Table 3.2) which selects a neighbor at random and the *Steepest Hill Climbing algorithm* (see Table 3.3) which chooses the closest node to the current solution.

Table 3.1: The Simple Hill Climbing Algorithm

Input: an individual *solution* representing the initial solution, termination criteria *term_crit*, the maximum number of neighbors n.

Output: the individual *sol* which represents the solution optimized by means of local search.

```
1: sol ← solution;
2: iter ← 0;
3: while (term_crit is not reached) do
4:     S ← getNeighbors(sol, n);
5:     new_sol ← getFirstBestNeighbor(S);
6:     if (evaluate(new_sol) < evaluate(sol)) then
7:         sol ← new_sol;
8:     end if
9:     iter ← iter + 1;
10: end while
11: return sol;
```

Table 3.2: The Stochastic Hill Climbing Algorithm

Input: an individual *solution* representing the initial solution, termination criteria *term_crit*.

Output: the individual *sol* which represents the solution optimized by means of local search.

```
1: sol ← solution;
2: iter ← 0;
3: while (term_crit is not reached) do
4:     new_sol ← getRandomNeighbor(sol);
5:     if (evaluate(new_sol) < evaluate(sol)) then
6:         sol ← new_sol;
7:     end if
8:     iter ← iter + 1;
9: end while
10: return sol;
```

Table 3.3: The Steepest Hill Climbing Algorithm

Input: an individual *solution* representing the initial solution, termination criteria *term_crit*, the maximum number of neighbors n.

Output: the individual *sol* which represents the solution optimized by means of local search.

```
1: sol ← solution;
2: iter ← 0;
3: while (term_crit is not reached) do
4:     S ← getNeighbors(sol, n);
5:     new_sol ← getBestNeighbor(S);
6:     if (evaluate(new_sol) < evaluate(sol)) then
7:         sol ← new_sol;
8:     end if
9:     iter ← iter + 1;
10: end while
11: return sol;
```

3.3.2 Simulated Annealing

Simulated Annealing (SA) is a stochastic computational technique that derived its name from the annealing process used to re-crystallize metals. As in the physical process, SA lets the solution to vary significantly while the temperature is high and fixes the changes as the temperature decreases, freezing it when the temperature reaches a value very near to 0. SA is one of the first metaheuristics that considers an explicit strategy to avoid local minima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (*uphill moves*) in order to escape from local minima [15].

A general pseudocode for SA is described in the Table 3.4. SA works by starting with an initial solution *sol*, and setting the temperature T to an initial (high) temperature T_0. At each iteration, SA randomly generates a neighbor *new_sol* of *sol* and compares its fitness value with the fitness value of the current solution *sol*. At this point, the neighbor solution *new_sol* is accepted as the new current solution if it is better than the current one or, in case it is worse, with a probability that is dependent on both the difference of fitness values *delta* and temperature T. The probability is generally computed by using the Boltzmann distribution $e^{\frac{-delta}{T}}$. Finally, SA reduces the temperature by following a predefined

Table 3.4: The Simulated Annealing Algorithm

Input: an individual *solution* representing the initial solution, the initial temperature T_0, the final temperature T_f.

Output: the individual *sol_best* which represents the solution optimized by means of local search.

```
1: sol ← solution;
2: sol_best ← solution;
3: T ← T_0;
4: while (T > T_f) do
5:    new_sol ← getRandomNeighbor(sol);
6:    delta ← evaluate(new_sol) − evaluate(sol);
7:    if (delta <= 0) then
8:       sol ← new_sol;
9:       if evaluate(new_sol) < evaluate(sol_best) then
10:         sol_best ← new_sol;
11:      end if
12:   else
13:      r ← getRandomValue();
14:      if r < e^(−delta/T) then
15:         sol ← new_sol;
16:      end if
17:   end if
18:   T ← updateTemperature(T);
19: end while
20: return sol_best;
```

cooling schedule. When the termination criteria are reached, SA returns the found best solution.

One critical point of the SA is the choice of an appropriate cooling schedule because it strongly affects the performance of the algorithm. In particular, in order to implement a cooling schedule, the following parameters should be specified:

- an initial temperature;
- a final temperature;
- a rule for decrementing the temperature.

In this research work, the cooling schedule proposed by Kirkpatrick [79] is used. In particular, Kirkpatrick suggested that a suitable initial temperature is

one that results in an average increase acceptance probability of about 0.8. The value of T_0 will clearly depend on the scaling of fitness function f and, hence, be problem-specific. It can be estimated by conducting an initial search in which all increases are accepted and calculating the average objective increase observed $\delta^- f^+$ on N trials. Therefore, in our implementation, the T_0 is computed by the following formula: $T_0 = \frac{\delta^- f^+}{ln(\chi_0)}$. With regard the final temperature T_f, it is necessary simply to choose a value very close to 0. The choice of a rule for decrementing the temperature is more complex. In literature, there are a lot of different methods. In our implementation, the *exponential cooling scheme* (ECS) proposed by Kirkpatrick et al. in [80] has been chosen. In general, this scheme reduces the current temperature by multiplying it with a constant α close to, but smaller than, 1. In particular, Kirkpatrick proposed $\alpha = 0.95$.

3.4 Population-based Search: Genetic Algorithms

The interest in Genetic Algorithms (GAs) began as early as the 1970s when Holland [67] first proposed them as search methods inspired by the mechanisms of evolutionary processes in nature. In particular, GAs are based on Darwinian principle of the natural selection which leads to the survival of the only fittest individuals. More in detail, in nature, evolution manifests itself as a succession of changes in species' features determined by genetic reproduction. The individual features, represented by chromosomes (formed in turn by genes), determine the survival capacity of individuals. Indeed, only fittest individuals capable of adapting to the changing environment survive and reproduce. Hence, the genes of the fittest individuals survive, while the genes of weaker one die out. Therefore, the evolution process is derived from the joint action of natural selection and the recombination of genetic material that occurs during reproduction and that generates diversity in the *gene pool* [119]. In detail, evolution is started when the genetic material from two parents recombines during reproduction. New combinations lead to new genes giving birth to a new gene pool. Precisely, the exchange of genetic material among chromosomes is named *crossover*. Segments of the

two parent chromosomes are swapped during crossover, arising the possibility of the "right" combination of genes for better individuals. Repeating selection and crossover operations leads to the continuous evolution of the gene pool and the generation of individuals that survive better in a competitive environment [119].

GAs try to solve an optimization (or search) problem by manipulating a *population* of potential solutions by means of reproducing the aforementioned natural evolution process. Specifically, they operate on encoded representations of the solutions, called *chromosomes*, that equivalent to the representations of individual features in nature. The encoding mechanism strongly depends on the nature of the problem variables. The algorithm evolution starts from a population of randomly generated individuals and consists in successive *generations*. In each generation, as in nature, a *selection process* provides the mechanism for selecting better solutions to survive. Each solution is evaluated by means a *fitness function* that reflects how good it is, compared with other solutions in the population. For maximum problem, the higher (the lower for minimum problem) is the fitness value of an individual and higher are its chances of surviving. Typically, GAs use the *roulette wheel selection scheme* [67]. It consists in giving to each chromosome a probability of being selected which is directly proportionate to its fitness score. In detail, if s_i is the fitness score of i^{th} individual of the population, its probability p_i of being selected is $p_i = \frac{s_i}{\sum_{k=1}^{N} s_k}$, where N is the number of individuals in the population. Therefore, the best solutions will have more possibilities of belonging to the next generated population. Instead, recombination of genetic material in GAs is simulated through two operators: *crossover* that exchanges portions between two randomly selected chromosomes and *mutation* that causes random alteration of the chromosome genes. In literature, there are different kinds of crossover. Traditionally, GAs use the *single-point crossover*. Precisely, this crossover operator randomly selects two chromosomes representing the *parents*. Then, it chooses a crossover point that can assume values in the range $[1, l-1]$, where l is the length of the chromosome. Each point of range has equal probability to be selected. The portions of the two chromosomes beyond this crossover point are swapped to form two new chromosomes, named *offsprings*. However, crossover is not always executed. As matter of fact, the algorithm performs crossover only if a randomly generated number in the range $[0, 1]$ is greater

than p_c, i.e., the *crossover rate*. Mutation, instead, runs through the genes in each of the chromosome in the population and mutates them in statistical accordance to the given mutation rate p_m. The genes of a chromosome are independently mutated, i.e, the mutation of a gene does not affect the probability of mutation of other genes. The algorithm evolution terminates when specified conditions such as the maximum number of generations or a specific fitness value are reached. The general GAs' structure is shown in Fig. 3.1.

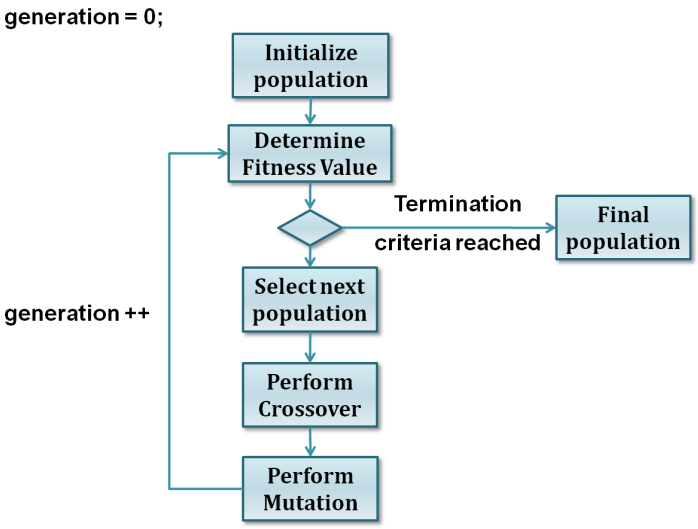


Figure 3.1: The typical steps of a genetic algorithm.

Thanks to their capability of exploring and exploiting promising regions of search space, GAs have the strong benefit not to prone to stalling at local optima. However, they can take relatively long time to locate the exact local optimum in a region of convergence (and may sometimes not to find the optimum with sufficient precision) [104]. For this reason, the idea to create an hybrid paradigm (see section 3.5) that combines genetic algorithms with the local search methods, known for their efficiency.

3.5 The Memetic Algorithms

Memetic Algorithms (MAs) are population-based meta-heuristic search methods inspired by both Darwinian principles of natural evolution (see section 3.4) and

Dawkins notion of a meme defined as a unit of cultural evolution that is capable of local refinements [29]. In detail, according to the philosophical theory of Richard Dawkins [29], human culture can be decomposed into simple units namely *memes*. Thus a meme is a "brick" of the knowledge that can be duplicated in human brains, modified, and combined with other memes in order to generate a new meme [99]. Within a human community, some memes are simply not interesting and will die away in a short period of time, whereas, other ones are relevant and, similar to an infection, will propagate within the entire community. An example of this concept is in the gossip propagation within human communities [99]. Some gossips are, *de facto*, more interesting than others and persist over time reaching all the individuals of the community. In addition, gossips can be subject to slight (or sometimes major) modifications. Sometimes these modifications make these gossips more interesting and thus more durable and capable to propagate. This example of life-time learning is also interesting in order to note the difference between memes and genes. In particular, the latter is not modified during the life-time of the individual, and is transmitted as they were inherited (of course, genetic information is mixed during sexual reproduction and can be subject to mutation as well, but this is a different process not alike to life-time learning). On the contrary, the former is much more plastic which also explains their comparatively faster rate of adaptation with respect to biological genes.

This charming interpretation of human culture inspired Moscato and Norman in late '80s [102] to define Memetic Algorithms (MAs). In practice, MAs blend together the most prominently ideas from local search techniques and population-based search, above all, genetic algorithms, by integrating local search processes within the global search successive generations in order to refine population individuals. In general, MAs structure can be seen as an iterated sequence of the following operations, aimed at converging a population of tentative solution toward an sub-optimal solution:

1. *Selection of parents*: as genetic algorithms, selection aims to determine the candidate solutions that will be used to create new solutions. Typically, selection for reproduction operates in relation with the fitness value of the candidate solutions;

2. *Application of Genetic operators*: recombination and mutation aims at creating new promising candidate solutions by blending existing solutions, a solution being promising if it can potentially lead the optimization process to new search areas where better solutions may be found;

3. *Update of the population*: this step decides whether a new solution should become a member of the population and which existing solution of the population should be replaced;

4. *Local improvement*: the goal of local improvement is to improve the quality of some or all individuals of the population. Candidate solutions undergo refinement which correspond the life-time learning of the individuals in the original metaphor of MAs.

The algorithm evolution terminates when it reaches a maximum number of iterations or a maximum of iterations without improvement or a specific target fitness, too. Representing the marriage between global search and local improvement, MAs have the complementary advantages of genetic algorithms (generality, robustness and global search efficiency) and local search methods (rapid convergence toward local minima). Without loss of generality, the template of MAs is summarized in listing 3.5.

MAs have been successfully applied, in recent years, to solve complex real-world problems (see section 3.5.1). However, despite their enormous benefits represented by more efficient search and convergence to higher quality solutions, MAs' performance strongly suffers the following issue [81]: *What is the best trade-off between local search and the global search provided by evolution?* In turn, this issue leads naturally to questions such as the following ones:

- *Local search frequency*: How often should local search be applied within the evolutionary cycle?

- *Order respect to genetic operators*: When should local search be applied?

- *Individual selection mechanism*: Which individuals in the population should be improved by local search?

Table 3.5: Template of Memetic Algotirthms.

Input: size of the population pop_size, crossover rate p_c, mutation rate p_m, termination criteria t, local search parameters l_p.

Output: the best chromosome $best\_chromosome$.

```
1: gen ← 0;
2: pop ← generateInitialPopulation(pop_size);
                    // Generate randomly an initial population pop of pop_size chromosomes
3: evaluateFitness(pop);                                  // Evaluate fitness value for each chromosome
4: while termination criteria t are not satisfied do
5:     offspring ← executeCrossover(pop, p_c);
                                    // Crossover chromosomes according to a crossover rate p_c
6:     offspring ← executeMutation(offspring, p_m);
                                    // Mutate chromosomes with a mutation probability p_m
7:     evaluateFitness(offspring);                        // Evaluate fitness value for new chromosomes
8:     pop ← selectPopulation(pop, offspring,pop_size);
                    // Select pop_size chromosomes to generate the next new population pop
9:     pop ← executeLocalSearch(pop, l_p);
                                    // Execute the local search refinement on population
10:    best_chromosome ← getBestChromosome(pop);
                                    // Select the best chromosome of the current population
11:    gen ← gen + 1;                                     // Increment number of iterations
12: end while
13: return best_chromosome;
```

- *Local search intensity*: How much computational effort should be allocated to each local search?
- *Local search method*: Which local search procedure should be used?

Typically, answering to these questions is very difficult. The common technique for giving answers is to perform an extensive phase of tuning aimed to carry out the best parameters suitable for a specific problem instance.

3.5.1 Applications of Memetic Algorithms

MAs have been shown to be efficient methods for solving several optimization problems [127]. Indeed, a lot of works have successfully applied MAs for solving well-known problems such as the classical Traveling Salesman Problem (TSP) [92][63], the job shop scheduling problem [48][27], the graph coloring problem [46][86], the non linear integer programming [124] and so on.

In addition, other combinatorial problems have also been addressed by MAs. For instance, in [131], the authors propose to apply MAs in wireless sensor net-

work (WSN) domain to solve the SET K-COVER problem. In detail, in order to extend the lifetime during which a WSN can cover all targets, an effective method is to partition the collection of sensors into several covers, each of which must include all targets, and then to activate these covers one by one. Since more covers enable longer lifetime, the SET K-COVER consists in finding the maximum number of covers. Another example is represented by [3], where the authors exploit MAs to solve an e-learning issue, modeled as the Plant Location Problem. In detail, this work presents a framework aimed at offering a set of personalised e-learning experiences (i.e. a structured collection of content and services able to facilitate learners in acquiring a set of competences about a specific domain) adapted to learner expectations. In order to achieve this goal, a convenient way to bind subjects, included in personalised learning paths, with learning activities (realized with learning services and learning objects) selected on the base of learner preferences is necessary. Therefore, the authors propose a multi-island memetic algorithm to face the binding problem formulated as a Plant Location Problem. Furthermore, in [1], a Tabu-based memetic algorithm that hybridizes a genetic algorithm with Tabu Search is presented as an improved method for course and examination timetabling problems. Finally, in [138], the authors propose a MA, which incorporates genetic algorithms with the variable neighborhood search algorithm, for the minimization of makespan in the heterogeneous multiprocessor scheduling problem. Other problems are vehicle routing [23][97], task allocation [64], maintenance scheduling problem [19][18].

However, MAs are not limited to solve combinatorial problems, but, they have successfully been also utilized in other fields. For example, they have been applied for training neural network [85][100], for selection of features in face recognition applications [82] and for analyzing microarray data [25].

3.6 A MAs' extension: Parallel Memetic Algorithms

As described in the previous section, MAs have been successfully used for a lot of applications. Indeed, they not only converge to high quality solutions, but

also search more efficiently than their genetic counterparts [127]. However, if, on the hand, the combination of global and local searches characterizing MAs leads to higher quality solutions, on the other hand, it involves an increment of the computational effort. One of the ways to face this problem is to develop parallel architectures for evolutionary optimization. For these reason, a variety of parallel memetic algorithm (PMA) models have also been studied recently [127]. PMAs extend the conventional MAs by introducing ideas belonging to the class of parallel genetic algorithms (PGAs). In general, PGA principle is to divide classical GA tasks across multiple processing nodes. This allows speeding up the search process and facilitating *speciation*, i.e., a process by which different subpopulations evolve in diverse directions simultaneously [127].

In literature, PGAs are categorized in three kinds: *master-slave PGAs*, *fine-grained PGAs* and *multiple-population* or *multiple-deme PGAs* [20]. In detail, master-slave PGAs evolve a single panmictic population without changing the traditional protocol of GAs, but, unlike conventional GAs, evaluations of individuals are distributed by scheduling fractions of the population among the processing slave nodes. Fine-grained PGA consists of a single spatially-structured population. Genetic operations such as selection and mating are limited to small groups of individuals, but group overlapping allows some interactions among all the individuals so that good solutions may disseminate across the entire population. Finally, multi-deme PGAs consists of different subpopulations which exchange individuals occasionally. This procedure is called *migration* and it is controlled by several parameters such as the number of individuals to be migrated or the kind of topology which determines the destination of migrants. Multi-deme PGAs are also known as *island parallel GAs* since the subpopulations can be viewed as relatively isolated demes. This classification can be naturally reported for PMAs.

The most important advantage of parallel MAs is that in many cases, the multi-population MAs provide better performance than single population-based algorithms, even when PMA is simulated sequentially, thanks to speciation phenomenon [72]. For this reason, PMAs are recognized as not only an extension of the traditional MA sequential model, but as a new class of algorithms in that they search the space of solutions differently [127].

3.7 Performance evaluation of search algorithms

In a scenario such as the ontology alignment problem, where a lot of algorithms for its resolution has been proposed, having techniques for understanding which approach is better than others is desiderata. In the case of the use of search algorithms, a comparison of their behaviour could be done attending to the efficiency and/or effectiveness criteria. However, when theoretical results are not available, it is necessary to focus on the analysis of empirical results [49]. In last years, statistical procedures are becoming more and more the most opportune methodology for performing this comparison. Statistical tests can be categorized in *parametric* and non-parametric techniques. Both classes of procedures have been used to compare algorithm behaviour. However, due to their constraints (for example, normal distribution of data samples), parametric statistical analysis could not be appropriate for analyzing algorithm behaviour [49]. Therefore, the current trend is to leave the comparison to the non parametric statistical tests.

In this section, we present a set of non parametric statistical procedures which have been used in literature and in this work to compare performances of different algorithms. In detail, we start with description of a non-parametric statistical procedure, named Wilcoxon's signed rank test [139], used for performing pairwise comparisons between two algorithms. Then, we describe two methods, named Friedman's test [47] and Holm's procedure [68], used for executing comparisons which include more than two different algorithms. Indeed, when we are interested in comparisons among several algorithms, pairwise statistical procedures such as Wilcoxon's test must not be used since repeating pairwise comparisons leads to an error which grows agreeing with the number of comparisons done, called *family-wise error rate* (FWER), defined as the probability of at least one error in the family of hypotheses [116].

3.7.1 Wilcoxon's signed rank test

As described in [49], the Wilcoxon's signed rank test is a non-parametric procedure employed in a hypothesis testing situation involving a design with two samples. It is a pairwise test used for answering this question: do two samples

represent two different populations? Therefore, it can be employed to detect significant differences between the behavior of two algorithms and so it is suitable for our experimentation: showing that the our approaches are better than existing ones for solving ontology alignment problem.

In general, a hypothesis testing as the Wilcoxon's test is a procedure in which sample data are employed to evaluate a hypothesis. In detail, in order to evaluate the research hypothesis, i.e., the statement of what a researcher predicts, two statistical hypotheses are necessary: the so-called *null hypothesis* (H_0) and the so-called *alternative hypothesis* (H_1). The null hypothesis is a statement of no effect or no difference and, in particular, for Wilcoxon's test is $H_0 : \theta_D = 0$, i.e, in the underlying populations represented by the two samples of results, the median of the difference scores equals zero. Instead, the alternative hypothesis represents a statistical statement indicating the presence of an effect or a difference. For Wilcoxon's test the alternative hypothesis can be $H_1 : \theta_D \neq 0$, but also $H_1 : \theta_D > 0$ or $H_1 : \theta_D < 0$ by considering directional hypothesis. Since the statement of a research hypothesis typically predicts the presence of a difference between two algorithms which are being studied, as result of an experimentation, the null hypothesis is expected to be rejected in favor of the alternative one.

In the following, we describe the test computation in a detailed way. Let N be the length of samples, i.e., the number of values (rows) that the samples contain and let d_i be the difference between the performance scores of the two algorithms on i^{th} value. The differences are ranked according to their absolute values. In case of ties, average ranks are assigned. Let R^+ be the sum of ranks for the rows on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored. Formally,

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{3.1}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{3.2}$$

Let T be the smallest of the sums, i.e., $T = min(R^+, R^-)$. If T is less than or

equal to the value of the distribution of Wilcoxon for N degrees of freedom (see Table B.12 in [144]), the null hypothesis of equality of means is rejected. Another evaluation of test results can be performed by considering the so-called p-value, the smallest level of significance that results in the rejection of the null hypothesis. The most common way for obtaining the p-value associated to a hypothesis is by means of normal approximations, that is, once computed the statistic associated to a statistical test or procedure, we can use a specific expression or algorithm for obtaining a z value, which corresponds to a normal distribution statistics. Then, by using normal distribution tables, we could obtain the p-value associated with z. The computation of the p-value in Wilcoxon's test follows this procedure. Since, the p-value provides information about whether a statistical hypothesis test is significant or not, and it also indicates something about "how significant" the result is: the smaller the p-value, the stronger the evidence against the null hypothesis. So, an experiment successfully ends if the computed p-value is small. For example, if the p-value is under the 0.01 value, it is possible to say that our test rejects null hypothesis at 1% significante level.

3.7.2 Friedman's test

Friedman's test is a non-parametric statistical procedure which aims at detecting if a significant difference among the behavior of two or more algorithms exists. In particular, under the null-hypothesis, it states that all algorithms are equivalent, hence, a rejection of this hypothesis implies the existence of differences among the performance of all studied algorithms [49].

The Friedman's test ranks the algorithms under comparison for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, and so on [31]. In case of tied data, the average of the ranks involved is assigned to all data tied for a given rank [116]. Formally, let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, i.e. $R_j = \frac{1}{N}\sum_i r_i^j$. The Friedman statistic (referred as χ_r^2) is approximated by means of the chi-square distribution with $k-1$ degrees

of freedom and is reported in equation 3.3.

$$\chi_r^2 = \frac{12N}{k(k+1)}[\sum_{j=1}^{N} R_j^2 - \frac{k(k+1)^2}{4}] \tag{3.3}$$

where k is the number of compared algorithms, N is the number of data sets, and R_j is the mean of the ranks for the j-th algorithm.

In order to reject the null hypothesis, the computed value χ_r^2 must be equal to or greater than the tabled critical chi-square value at the specified level of significance [116].

3.7.3 Holm's test

Holms procedure is a multiple comparison procedure that works by setting a control algorithm and comparing it with the remaining ones. Normally, the algorithm which obtains the lowest value of ranking in the Friedman's test is chosen as control algorithm. Holm's test works on a family of hypotheses where each one is related to a comparison between the control method and one of the remaining algorithms. In details, the test statistic for comparing the i^{th} and j^{th} algorithm is reported in equation 3.4.

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6N}} \tag{3.4}$$

The computed z value is used to find the corresponding probability from the table of the normal distribution (the so-called p-value), which is then compared with an appropriate level of significance α [31]. In order to perform its evaluation, Holm's method sequentially checks the hypotheses ordered by their significance. In details, it orders the p-values by denoting them as $p_1, p_2, \ldots, p_{k-1}$ so that $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. Then, it compares each p_i with $\alpha/(k-i)$ starting from the most significant p. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we are allowed to compare p_2 with $\alpha/(k-i)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well [31].

Chapter 4

MemeOptiMap: A Memetic Optimization System for the Ontology Alignment

In the previous chapters, we have discussed the ontology alignment problem, i.e., the research issue that we want to address, and the memetic algorithms, i.e., the methodology that we investigate to face the problem at hand. In this chapter, we describe our first contribution to ontology alignment researches consisting in designing and implementing a memetic algorithm-based ontology alignment system, named *MemeOptiMap* (see section 4.2). In order to achieve this result, we have reformulated the ontology alignment problem as an optimization one (see section 4.1). Since implementing an efficient memetic algorithm requires to choose a suitable configuration of its parameters, in designing our system, we have investigated different settings in order to find the best local configuration (see section 4.3). Given the computational cost of the first version of *MemeOptiMap*, we have designed its parallel extension (see section 4.4).

4.1 The Ontology Alignment as Optimization Problem

This thesis aims at investigating the application of memetic algorithms to the ontology alignment problem. Our first idea, presented in [2] [7], is to reformulate the ontology alignment problem as an optimization problem, and, consequently, to design an efficient memetic algorithm for solving it. In detail, the formulated ontology alignment optimization problem considers as solution the set of mapping elements minimizing the distance (or, equivalently, maximizing the similarity) among the entities belonging to ontologies under alignment. This result is achieved by exploiting an objective function that assesses the quality of an alignment by summing the *distances* between ontologies entities composing mapping elements of alignment at hand. These distances are computed by using a *suitability function* able to simultaneously evaluate lexical, linguistic and structural distances between the mapping element entities and aggregate them in an overall distance value by means of a so-called *aggregation strategy*.

Formally, by considering the definitions 1 and 4 related to, respectively, ontology and alignment, the ontology alignment process can be formulated as an optimization problem as depicted by the Definition 8. *MemeOptiMap* exploits this definition for implementing an efficient research approach capable of computing a sub-optimal ontology alignment and achieving better performances than other approaches.

Definition 8 (Alignment Optimization Problem) *The alignment optimization problem is a quadruple* (O_1, O_2, A_{set}, F) *where:*

- O_1 *and* O_2 *are the ontologies to align (the problem instance);*
- A_{set} *is the set of all possible alignments between* O_1 *and* O_2 *(the set of feasible solutions);*
- $F : A_{set} \rightarrow \mathbb{R}$ *is the objective function (to be minimized) evaluating the*

quality of an alignment $A \in A_{set}$ *as follows:*

$$F(A) = \sum_{i=1}^{|A|} f(c_i) \text{ with } c_i \in A \tag{4.1}$$

where $f : A \rightarrow [0, 1]$ *is the* suitability function *which associates a value in the* $[0, 1]$ *interval to each mapping element* c_i *in the alignment* $A \in A_{set}$. *The function* f *is used to determine the goodness of a mapping element to achieve an optimal alignment. The function* f *is computed by aggregating, in a weighted way, a collection of similarity measures:*

$$f(c_i) = \phi(\overrightarrow{s}(c_i), \overrightarrow{w}) \tag{4.2}$$

where the function ϕ *represents an aggregation strategy which combines the vector of similarity measures* $\overrightarrow{s}$ *by considering a weights vector* $\overrightarrow{w}$.

A list of possible similarity measures and aggregation strategies useful to implement the suitability function f is reported, respectively, in sections 2.3.1.2 and 2.3.1.3. For sake of clarity, in this research work, only distance-based similarity measures are taken in account and, consequently, a lower value of $F(A)$ corresponds to an alignment A nearer to optimal one. Therefore, the considered alignment problem is a *minimum optimization problem*.

4.2 *MemeOptiMap* System

This section aims at presenting a new system able to perform an automatic matching process based on an emergent hybrid evolutionary approach, named Memetic Algorithms (MAs). As described in section 3.5, MAs are population-based search methods which combine evolutionary algorithms (EAs) with local search (LS) methods. The name is inspired by Richard Dawkins' concept of a *meme*, which represents a unit of cultural evolution that can exhibit local refinement [29]. Indeed, the MAs have the capability of realizing local search processes within the successive generations characterizing global search methods in order to refine population individuals. In particular, we design a memetic algorithm

which extends a genetic algorithm with the stochastic hill climbing search. In detail, the designed memetic algorithm based ontology alignment system, named *MemeOptiMap*, takes in input two ontologies O_1 and O_2, and gives in output a suboptimal alignment A as follows. Initially, the system randomly generates a population of possible alignments between the input ontologies O_1 and O_2; in detail, the algorithm generates a collection of chromosome genes (see Fig. 4.1) by using an uniform probability distribution. At each iteration, the algorithm computes new alignments by evolving the current population by means of traditional genetic operators: the *crossover* and *mutation* operators. For each iteration, the number of applications of crossover is determined by the crossover rate value p_c; the crossover operands are randomly selected from the current population by means of a uniform probability distribution. At the same way, for each chromosome belonging to the current population, the number of gene mutations is determined by the mutation rate value p_m. After these conventional genetic steps are computed, the system performs a stochastic hill climbing search in order to refine the best alignment belonging to the current genetic population. Successively, a genetic selection operator such as the roulette wheel is applied in order to generate a new alignment population used by the system to start the next iteration. At the end of each iteration, if a reference alignment has been provided, precision, recall and f-measure of the best alignment are computed and stored for test goals. The system ends its evolution when the termination criteria are satisfied. The behavior of *MemeOptiMap* is summarized in pseudocode reported in Table 4.1.

Hereafter, a detail description of each single component of *MemeOptiMap* followed by a discussion on matching dimensions and some implementative details is given. The section ends with some experimental results.

4.2.1 Basic components of *MemeOptiMap*

The components of *MemeOptiMap* which require a more detail description are:

- the chromosome structure used to represent the solution of our problem, i.e, an alignment;

Table 4.1: Ontology Alignment Memetic Algorithm.

Input: two ontologies O_1 and O_2 to align; GA parameters (size of the population *pop_size*, crossover rate p_c, mutation rate p_m), termination criteria *term_crit*; local search parameters ls_p.

Output: the best optimized alignment (represented by the final best chromosome) between the ontologies O_1 and O_2.

```
1: gen ← 0;
2: pop ← generatePopulation(pop_size);
                    // Generate randomly an initial population pop of pop_size chromosomes
3: evaluateFitness(pop);                          // Evaluate fitness value for each chromosome
4: best_chromosome ← getBestChromosome(pop);
                                  // Select the best chromosome of the current population
5: quality ← evaluateAlignment(best_chromosome);
                                                  // quality contains the information about
                    // the computed conformance measures on the current best chromosome
6: while (term_crit are not satisfied) do
7:     executeCrossover(pop, p_c);        // Crossover chromosomes according to a crossover rate p_c
8:     executeMutation(pop, p_m);          // Mutate chromosomes with a mutation probability p_m
9:     evaluateFitness(pop);                      // Evaluate fitness value for new chromosomes
10:    best_chromosome ← getBestChromosome(pop);
                                  // Select the best chromosome of the current population
11:    local_chr ← executeLocalSearchMethod(best_chromosome, ls_p);
                                // Execute the local search process on the best chromosome
12:    executeSelection(pop, pop_size);
                        // Select pop_size chromosomes to generate next new population pop
13:    gen ← gen + 1;                                  // Increment number of iterations
14:    quality ← evaluateAlignment(local_chr);        // quality contains the information about
                    // the computed conformance measures on the current best chromosome
15: end while
16: return best_chromosome;
```

- the employed fitness function which allows the evaluation of the considered solutions;
- the integrated local search process.

4.2.1.1 The alignment chromosome structure

Let consider that each ontology entity is enumerated from 0 to $n + m + k - 1$ where m, n, k are, respectively, the cardinalities of the sets C, P, I which compose an ontology (see definition 1). A chromosome representing an alignment A is a integer vector where each cell (gene) represents a correspondence between an entity of the first ontology and any entity of the second one. More in detail, by considering that the i^{th} vector cell contains the integer j, the cell represents the correspondence (e_i, e_j) where e_i is the i^{th} entity of the first ontology and e_j is the

j^{th} entity of the second one. Given this chromosome structure, its size does not depend on the cardinality of the second ontology at all, but, precisely, it is equal to the number of entities belonging to the first ontology. Formally:

Definition 9 (Alignment Chromosome) *A chromosome S representing an alignment A between two ontology O_1 and O_2 is the set:*

$$S = \{(e_0, e_{j_0}), (e_1, e_{j_1}), (e_2, e_{j_2}), \ldots, (e_h, e_{j_h})\}$$

where $h = |O_1| - 1$ and $j_l \in \{0, 1, 2, \ldots, |O_2| - 1\}$ with $l = 0, 1, 2, \ldots, h$.

Therefore, the chromosome dimension depends on alignment problem instance.

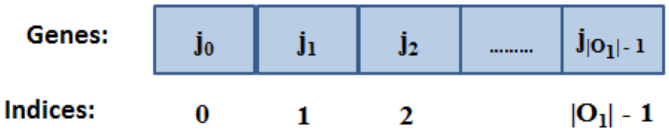


Figure 4.1: The general structure of an alignment chromosome: each gene represents the correspondence (e_i, e_{j_i}) where $i = 0, 1, 2, \ldots, |O_1| - 1$ and $j_i \in \{0, 1, 2, \ldots, |O_2| - 1\}$

In order to better understand the alignment chromosome structure, let us consider the two ontologies O_1 and O_2 presented in Figs. 2.2 and 2.3 whose entities are indexed as shown in Fig. 4.2 and the possible alignment chromosome depicted in Fig. 4.3-a). The resulting alignment is shown in Fig. 4.3-b).

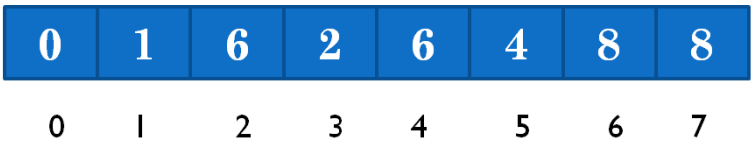
It is worth noting that, due to the designed chromosome structure, *MemeOptiMap* builds alignments characterized by a cardinality $(n : 1)$, i.e., an entity of the first ontology can be associated with an only entity of the second one, whereas, an entity of the second ontology can be associated also with more entities of the first one.

4.2.1.2 Fitness function

Section 4.1 is devoted to formulate the ontology alignment process as a minimization problem based on the evaluation of the objective function provided by

Ontology O_1	Indexes of entities	Ontology O_2	Indexes of entities
Object	0	thing	0
vehicle	1	conveyance	1
ship	2	car	2
car	3	Volkswagen	3
speed	4	Porsche	4
owner	5	motor	5
Has_speed	6	speed	6
Belongs_to	7	hasMotor	7
		hasProperty	8

Figure 4.2: The ontologies O_1 and O_2 whose the entities are indexed by 0 to cardinality of the ontology minus one



a)

Ontology O_1	Ontology O_2
Object	thing
vehicle	conveyance
ship	speed
car	car
speed	speed
owner	Porshe
Has_speed	hasProperty
Belongs_to	hasProperty

b)

Figure 4.3: In a) a possibile alignment chromosome for the ontology O_1 and O_2 and in b) the corresponding alignment.

Eq. (4.1). As a consequence, *MemeOptiMap* exploits the same function for evaluating the fitness values of the chromosomes composing the population of solutions evolved by the designed memetic algorithm.

4.2.1.3 The integrated local search process

In order to design a *competent* [55] MA, a lot of issues must be addressed over the integrated local search process as described in section 3.5.

Therefore, in order to complete the description of *MemeOptiMap*, the aforementioned issues must be faced. In detail, *MemeOptiMap* is characterized by the following features:

- *Local search frequency = 1*, i.e., the local search is applied within each evolutionary cycle;
- *Order respect to genetic operators = after*, i.e., local refinement is executed after crossover and mutation operators;
- *Individual selection mechanism = only the best*, i.e., only the best chromosome of population is improved by the local search process;
- *Local search intensity = equal to n local search iterations*, i.e., each local search process ends after running n iterations.
- *Local search method = stochastic hill climbing*, i.e., the selected local search method is the stochastic version of the Hill Climbing search (see section 3.3).

4.2.2 Discussions on Matching dimensions

Beside the general scheme presented in Definition 2, it is useful to consider a collection of additional features related to the ontology alignment process, known as *dimensions* (see section 2.3.1.1). They represent constraints or restrictions on the ontology alignment process which influence the behavior of ontology alignment systems and, as a consequence, they can be used for performing their classification [43]. This section is devoted to present the dimensions and the corresponding

values which characterize the behavior of *MemeOptiMap*. This description allows to highlight the features of the produced alignments, and, at the same time, categorize *MemeOptiMap* in the state of the art.

By analyzing the features and the behaviour of *MemeOptiMap*, it is characterized by the following matching dimensions (see section 2.3.1.1):

- *languages*: The input ontologies are coded by OWL;
- *complete/update*: a whole ontology alignment is computed from scratch. Therefore, *MemeOptiMap* does not take in input an initial partial alignment or in other words, the initial alignment is equal to *emptyset*;
- *Resources*: *MemeOptiMap* exploits WordNet as dictionary for the linguistic similarity computation;;
- *Proper parameters*: A collection of parameters are exploited for controlling the behavior of the memetic algorithm used by *MemeOptiMap* for generating the ontology alignment;
- *multiplicity*: $(n : 1)$ due to the designed chromosome structure;
- *relations of output alignment*: currently, *MemeOptiMap* takes into account only equivalence relations.

Moreover, as for the high level classification in schema or instance-based approach [43], *MemeOptiMap* is hybrid since it can exploit similarity measures considering both schema and instance-based information.

4.2.3 Implementative details

MemeOptiMap has been implemented in Java[1]. It is mainly based on two Java libraries:

- Alignment API[2] which serves as an interface to ontologies and alignments;

[1] http://www.java.com/en/
[2] http://alignapi.gforge.inria.fr/

Table 4.2: Parameters of *MemeOptiMap*

Kind	Name	Description
Genetic parameters	population	the number of chromosomes
	crossoverRate	the probability of crossover operator
	mutationRate	the problability of mutation operator
	termination	the termination criteria to be chosen among number of iterations, number of fitness evaluations, convergence, precision, recall, F-measure
Local search parameters	intensity	the number of iterations performed by local search
	Local search selection mechanism	which and how many chromosomes are selected for local refinement
	method	the local search method
Ontology alignment parameters	matchers	the used set of similarities measures
	aggregation	the used aggregation strategy
	threshold	the value used to establish the validity of a computed correspondence
	vector of weights	the set of weights used in the aggregation step

- JGap[1] used to implement genetic components such as chromosomes, genetic operations and so on.

As for the local search methods, we have designed a custom implementation. *MemeOptiMap* setting can be configured by a parameter file. The complete list of parameters is reported in table 4.2.

4.2.4 Experimental results

In this section, in order to investigate the performances of *MemeOptiMap* and, as a consequence, show the suitability of memetic algorithms in the ontology alignment context, a set of experiments is performed. In detail, *MemeOptiMap* is used to align ontologies belonging to some test cases of the well-known benchmark dataset provided by the OAEI (see section 2.3.2.1) shown in Table 4.3. According to OAEI policies, the benchmark reference alignments take into account only the

[1]http://jgap.sourceforge.net/

matching, respectively, between ontology classes and properties.

Table 4.3: Benchmarks descriptions

Identifier	**Variant features**
101	the ontology itself
103	a generalisation in OWL Lite
104	a restriction in OWL Lite
204	different naming conventions
205	the labels are replaced by synonyms and the comments have been suppressed
208	some labels are in capital letters
221	all subclass assertions to named classes are suppressed
222	a hierarchy still exists but has been strictly reduced
223	numerous intermediate classes are introduced within the hierarchy
224	all individuals have been suppressed from the ontology
225	all local restrictions on properties have been suppressed from the ontology
228	properties and relations between objects have been completely suppressed
229	Some classes have become instances
230	Some components of classes are expanded in the class structure (flattening entities)
232	no hierarchy and no instance
233	no hierarchy and no property
236	no property and no instance
237	flattened hierarchy + no instance
238	expanded hierarchy + no instance
241	no hierarchy + no instance + no property

The configuration characterizing *MemeOptiMap* during the experimental session has been set as shown in Table 4.4.

In order to complete the description of experiments, the detail about the hardware configuration used to run *MemeOptiMap* is provided:

- Processor: Intel Core i5;
- CPU Speed: 2.3 GHz;

Table 4.4: Parameter Configuration

Parameter	**Value**
Population size	20 chromosomes
Crossover rate	0.8
Mutation rate	0.02
Local search intensity	300 iterations
Local search individual selection mechanism	Only the best chromosome
Local search method	Stochastic Hill Climbing
Termination condition	5000 evaluations of fitness
Similarity Measures	*Entity Name Distance Matcher* *Entity Comment Distance Matcher* *Super Hierarchy Distance Measure* *Word Net Synonymy Name Distance Measure* *Domain and Range Restrictions Distance Measure*
Aggregation	OWA operator
Weights	[0.4, 0.3, 0.15, 0.1, 0.05]
Threshold	0.0

- RAM Capacity: 4GB.

The performances yielded by *MemeOptiMap* are assessed by means of standard evaluation measures considered by OAEI: *precision* (see definition 5), *recall* (see definition 6) and *F-measure* (see definition 7). Table 4.5 shows the results of the experiments. In particular, the reported values represent the average on ten runs.

As shown in Table 4.5, *MemeOptiMap* achieves good results in all considered test cases. Therefore, these preliminary experiments highlight the suitability of our proposal. However, there are still relevant margins of improvement which could be obtained by better tuning the configuration parameters both as for algorithm parameters (e.g. termination criteria, local search intensity, local search method, etc.) and ontology alignment parameters (e.g. similarity measures, weights, etc). For this reason, in the next section, we prove to look for the best local configuration for *MemeOptiMap*.

Table 4.5: Experimental results for *MemeOptiMap*

No.	***precision***	***recall***	***F-measure***
101	0,72	0,72	0,72
103	0,73	0,73	0,73
104	0,75	0,75	0,75
204	0,69	0,69	0,69
205	0,64	0,64	0,64
208	0,62	0,62	0,62
221	0,75	0,75	0,75
222	0,73	0,77	0,75
223	0,67	0,67	0,67
224	0,73	0,73	0,73
225	0,76	0,76	0,76
228	0,98	0,98	0,98
230	0,57	0,77	0,66
232	0,75	0,75	0,75
233	0,98	0,98	0,98
236	0,99	0,99	0,99
237	0,74	0,77	0,75
238	0,68	0,68	0,68
241	0,97	0,97	0,97

4.3 Looking for the best local configuration for *MemeOptiMap*

The memetic algorithms (MAs) are population-based optimization methods which combine genetic algorithms with local search processes. As similar to genetic algorithms, MAs try to solve a search problem by evolving an initial random population of solutions by means of genetic operators, such as *crossover* and *mutation*, and local search refinements. Therefore, thanks to the marriage between global search and local improvement, MAs are characterized by a rapid convergence, like local search methods, but, differently from these, they not to prone to stalling at local optima due to their capability of exploring and exploiting promising regions of search space provided by the genetic component.

Nevertheless, in spite of their potential benefits, it is strongly accepted that the efficiency of MAs is affected by the following issue [81]: *What is the best trade-off between local search and the global search provided by evolution?* Typically, answering to this question is very difficult and depends on the specific problem at hand. Therefore, in [10], we have investigated the combination between genetic algorithms and different local search methods in order to find the best trade-off which improves performances of *MemeOptiMap*. The research work has involved a particular scenario: the communication in a multi-agent system. In this section, we present the work done in this research area.

In detail, we investigate different versions of *MemeOptiMap* system obtained by combining genetic algorithms and different local search methods. In particular, the explored local search algorithms are: two variants of the Hill Climbing Search, i.e., the Simple Hill Climbing Algorithm and the Steepest Hill Climbing in addition to the already explored Stochastic Hill Climbing, and the Simulated Annealing. See section 3.3 for a detailed description of these local search methods. The corresponding versions of system *MemeOptiMap* will be referred as MSHC, MStoHC, MSteHC and MSA which represent the combination of genetic algorithms with, respectively, the Simple Hill Climbing Algorithm, the Stochastic Hill Climbing Search, the Steepest Hill Climbing Search and the Simulated Annealing. By an implementative point of view, these versions are generated by replacing the call of function **executeLocalSearchMethod()** (see Table 4.1) with the specific local search method.

A set of experiments have been performed in order to investigate the performances of different versions of *MemeOptiMap* in a multi-agent system scenario by trying to define the best local configuration capable of improving agent interoperability.

Hereafter, the precise details about experiments and the explored multi-agent case study are provided.

4.3.1 Experimental results

The four different versions of *MemeOptiMap* (MSHC, MStoHC, MSteHC and MSA) are compared by using some test cases of the well-known benchmark

dataset provided by the OAEI (see section 2.3.2.1) and shown in Table 4.3. In order to compare the performances of the all proposed versions, each system has been run for a limited time represented by the achievement of a number of fitness evaluations. Once all systems have computed their alignments, the correspondent F-measure value (in percentage) is used as feature to compare the quality of the produced alignments. In detail, the greater the F-measure value and the better the system performances.

The common parameters characterizing the behavior of the four considered systems during the experimental session are shown in Table 4.6.

Table 4.6: Parameter Configuration

Parameter	**Value**
Population size	20 chromosomes
Crossover rate	0.8
Mutation rate	0.02
Local search individual selection mechanism	Only the best chromosome
Termination condition	5000 evaluations of fitness
Similarity Measures	*Entity Name Distance Matcher* *Entity Comment Distance Matcher* *Super Hierarchy Distance Measure* *Word Net Synonymy Name Distance Measure* *Domain and Range Restrictions Distance Measure*
Aggregation	OWA operator
Weights	[0.4, 0.3, 0.15, 0.1, 0.05]
Threshold	0.0

Instead, the particular parameters characterizing the single systems are the following ones:

- The MHC algorithm's parameters are:
  - maximum number of iterations = 100
  - maximum number of neighbors = 30
- The MStoHC algorithm's parameters are:

- maximum number of iterations = 300

- The MSteHC algorithm's parameters are:
  - maximum number of iterations = 200
  - maximum number of neighbors = 30

- The MSA algorithm's parameters are:
  - number of trials = 20
  - final temperature = 0.00000005

These parameters were computed in an empirical way by performing some preliminary experiments and trying to get the best configuration for each particular algorithm.

In order to complete the description of experiments, details about the hardware configuration used to run the systems are provided:

- Processor: Intel Core i5;
- CPU Speed: 2.3 GHz;
- RAM Capacity: 4GB.

The comparison among the considered systems (MHC, MStoHC, MSteHC, and MSA) is formally carried out by means of a multiple comparison procedure which consists in two steps: in the first one, a statistical technique such as the Friedman's test (see section 3.7.2) is used to determine whether the results provided by the considered algorithms present any inequality; in the second one, which is performed only if in the first step an inequality is found, a post-hoc test such as Holm's test (see section 3.7.3) is led in order to determined which algorithm better outperforms.

As described in section 3.7.2, Friedman's test is a non-parametric statistical procedure which, under the null-hypothesis, states that all compared algorithms are equivalent. Hence, a rejection of the null hypothesis implies the existence of differences among the performance of all studied algorithms [49]. The rejection

Table 4.7: Samples for Friedman's test. Each value represents the average of F-measure values (in percentage) on 20 runs. Among round parentheses, there is the computed rank of each system for benchmark at hand.

benchmark number	V_{MSHC}	V_{MStoHC}	V_{MSteHC}	V_{MSA}
101	77,43 (1)	71,93 (2)	71,13 (3)	63,46 (4)
103	77,09 (1)	72,51 (2)	72,05 (3)	64,49 (4)
104	79,72 (1)	75,03 (2)	70,22 (3)	66,21 (4)
204	72,39 (1)	69,3 (2)	68,5 (3)	61,86 (4)
205	67,7 (1)	63,46 (2)	60,94 (3)	55,33 (4)
208	69,19 (1)	62,43 (3)	64,15 (2)	57,39 (4)
221	79,15 (1)	74,69 (2)	69,87 (3)	66,67 (4)
222	78,83 (1)	74,85 (2)	69,94 (3)	66,67 (4)
223	68,84 (1)	67,24 (2)	62,89 (3)	55,67 (4)
224	78,7 (1)	73,2 (2)	71,59 (3)	68,5 (4)
225	78,46 (1)	75,49 (2)	72,85 (3)	67,12 (4)
228	98,32 (2)	97,64 (3)	98,65 (1)	90,9 (4)
230	68,38 (1)	65,75 (2)	62,46 (3)	58,51 (4)
232	78,81 (1)	75,37 (2)	68,73 (3)	65,06 (4)
233	96,97 (3)	97,64 (2)	98,99 (1)	91,58 (4)
236	98,99 (1)	98,65 (2)	97,31 (3)	92,59 (4)
237	77,31 (1)	75,32 (2)	69,94 (3)	67,72 (4)
238	70,1 (1)	67,93 (2)	63,57 (3)	55,55 (4)
241	96,3 (3)	97,31 (2)	97,98 (1)	92,59 (4)
averages	79,615 (1,26)	76,618 (2,10)	74,303 (2,63)	68,835 (4,00)

of the null hypothesis occurs when the computed value χ_r^2 is equal to or greater than the tabled critical chi-square value at the specified level of significance [116]. In our experimentation, a level of significance α equal to 0.05 is chosen. The samples V_{MHC}, V_{MStoHC}, V_{MSteHC} and V_{MSA} used for the Friedman's test are presented in Table 6.5. For each system, the sample is obtained by performing the average of F-measure values (in percentage) on the 20 runs.

By performing the Friedman's test, the computed χ_r^2 value is $45,06$. Since in our case $k = 4$, our analysis has to consider the critical value $\chi_{0,05}^2$ for three degrees of freedom that is equal to $7,82$. Since the computed $\chi_r^2 = 45,06$ value

Table 4.8: Holm's test

i	System	z value	unadjusted p-value	$\alpha/(k-i)$, $\alpha = 0,05$
1	MSA	6,5417	$6,0823 \cdot 10^{-11}$	0,00167
2	MSteHC	3,2708	0,0011	0,025
3	MStoHC	2,0055	0,0449	0,0500

is greater than its associated critical value $\chi^2_{0,05} = 7,82$, the null hypothesis is rejected and it is possible to assess that there is a significant difference between at least two of the four compared systems.

Attending to this result, a post-hoc statistical analysis is needed to conduct pairwise comparisons in order to detect concrete differences among compared algorithms. In our experimentation, we use Holm's procedure. This test is a multiple comparison procedure that works by setting a control algorithm and comparing it with the remaining ones. Normally, the algorithm which obtains the lowest value of ranking in the Friedman's test is chosen as control algorithm. In our case, as shown in Table 6.5, the system with the lowest value of ranking is MHC. As described in section 3.7.3, Holm's test works on a family of hypotheses where each one is related to a z-value corresponding to the comparison between the control method and one of the remaining algorithms. The computed z value is used for finding the corresponding probability from the table of the normal distribution (the so-called p-value), which is then compared with an appropriate level of significance α [31], in our experimentation equal to 0.05. All data computed by the Holm's procedure are depicted in Table 6.7. By analysing data, Holm's procedure rejects all hypothesis. As a consequence, it is possible to state that the MHC statistically outperforms better than MSA, MStoHC and MSteHC at 5% significance level.

4.3.2 Case study: Agent Communication

Interoperability is a crucial problem in multi-agent systems where autonomous artificial entities have to interact and cooperate in order to achieve a common goal. In the last years, ontologies are become an essential tool for enabling interoperability by allowing communication and exchanging information and services

within agents environments. In particular, agents use the ontologies as a common way to represent their "view of the world". However, ontological representation of knowledge could not be sufficient to achieve high levels of interoperability because various agents involved in a given information exchange, may potentially use different ontologies to represent the same domain of interest.

The simple idea to solve this problem may be to force agents to interact by exploiting a common ontology. However, open environments (where a central design is neither possible nor desirable) populated with heterogeneous agents make the common ontology case unfeasible [45]. Moreover, a typical attitude of the enterprises is the refusal to convert all the content of their ontologies if the target ontology is less expressive or not considered as a *de facto* standard [45]. Therefore, the most solid solution for enabling a real agent interoperability is to perform an ontology alignment process to lead proprietary ontologies into a mutual agreement. The quality of an alignment directly affects the interoperability: a more accurate alignment results in a more efficient agent interoperability.

The results of Holm's test presented in previous section states that the version of *MemeOptiMap* which combines genetic algorithms and Hill Climbing (MHC system) shows better performances than the other considered approaches. The aim of this section is to quantify the improvement provided by MHC system in terms of agent interoperability. In order to achieve this aim, the average F-measure values shown in Table 6.5 are exploited for the interoperability evaluation. Indeed, as already mentioned, a more accurate alignment allows agents a more efficient communication and exchange of information. In particular, the percentage improvements between the average F-measure values related to the MHC system and the other methods are computed. The results state that MHC system improves agent interoperability of 4%, 7% and 15%, respectively, with respect to MStoHC, MSteHc and MSA systems. By taking in account the high amount of messagges exchanged within an agent environment, these percentage improvements represent considerable benefits.

4.4 A parallel extension of *MemeOptiMap*

The memetic algorithms (MAs) are population-based optimization methods which integrate genetic algorithms with local refinement and, consequently, increase the convergence speed of the evolutionary process. However, if, on the hand, the combination of global and local searches leads to higher quality solutions, on the other hand, it involves an increment of the computational effort. Therefore, in order to face this issue, we have developed a parallel extension of *MemeOptiMap* [11]. This choice is supported by recent studies that have shown how parallel memetic algorithms (PMAs) converge to high quality solutions significantly faster than canonical parallel genetic algorithms [32] and MAs [72]. In literature, there exists a lot of PMA models (see section 3.6). Among them, in our research work, we choose to implement a so-called multi-island parallel memetic algorithm. This parallel optimization technique 1) evolves a collection of subpopulations to lead the search process in diverse directions simultaneously and 2) opportunely migrates individuals among subpopulations with aim of restoring diversity and preventing premature convergence to a low-quality solution. The choice of this kind of parallel strategy is due to its advantages derived from the use of semi-isolated populations which helps preserving diversity and increases the chances of escaping from local optima: an issue affecting more memetic algorithms than genetic counterpart.

In order to implement a parallel version of *MemeOptiMap*, in our research work, we exploit a Multi-Agent System (MAS) paradigm. MASs allow building distributed systems, where it is assumed that the computational components, named *agents*, are autonomous, i.e., able to control their own behaviour in the furtherance of their own tasks [141], and interacting for achieving a common objective. Thanks to the collaborative agents' behavior, a MAS is capable of providing different design benefits such as parallelism, robustness, scalability, geographic distribution and cost effectiveness [30]. These features have led to use the idea of collaborative agents in order to implement a distributed version of *MemeOptiMap* based on a multi-island parallel memetic algorithm.

Hereafter, more details about the designed parallel version of *MemeOptiMap* and some experimental results.

4.4.1 Architecture

The parallel version of *MemeOptiMap* is based on a multi-island parallel memetic algorithm implemented through a multi-agent system. In detail, the designed multi-agent system (see Fig. 4.4) is composed of two kinds of agents: a *coordinator agent* and several *optimizing agents* organized in a ring way. The aim of the coordinator agent is to start the ontology matching process on a pair of ontologies, randomly create the initial subpopulations (collections of chromosomes where each chromosome represents a candidate ontology alignment) and assign each one of them to a given optimizing agent. Then, the coordinator agent waits until optimizing agents end their tasks. Each optimizing agent performs, simultaneously with other optimizing agents, a memetic optimization process to produce an ontology alignment. The memetic optimization process performs two steps: 1) the generation of new individuals through the application of the traditional genetic operators such as single-point crossover and mutation and 2) the replacement of worst chromosomes of the genetic population with the best individuals refined through a local search procedure. All new individuals are evaluated through an appropriate fitness function. The new population is obtained by using a genetic selection operator such as the roulette wheel. During its activity, each optimizing agent checks if the migration moment is achieved. In that case, an optimizing agent executes all tasks related to migration procedure such as sending chromosomes to and receiving new ones from a neighbor optimizing agent. When an optimizing agent ends its evolution due to the achieving of specific termination criteria, it sends the best chromosome of its subpopulation to coordinator agent. When coordinator agent receives a solution from each optimizing agent, it computes the best solution and returns it in output. The behaviors of the coordinator agent and an optimizing one are, respectively, summarized in Table 4.9 and 4.10.

The fundamental elements of each optimizing agent necessary for performing a memetic optimization process and cooperate with other optimizing agent are: 1) the chromosome structure representing an alignment, 2) the employed fitness function that allows the evaluation of the considered solutions, 3) the several issues about the integrated local search process, 4) the migration process.

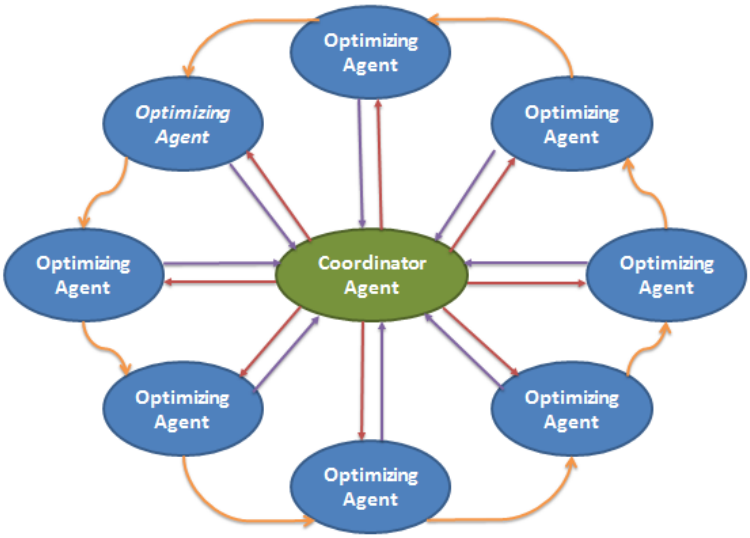


Figure 4.4: The architecture of *MemeOptiMap* based on an island parallel memetic algorithm implemented through collaborative agents

As for the fitness function, it is the same used by the sequential version of *MemeOptiMap* described in section 4.2.1.2. As for the other three first points, the chromosome structure, the features related to the integrated local search process and the migration process, they require a description of more details.

As described in section 4.2.1.1, a chromosome representing an alignment is defined as a integer vector where each cell individuates the pair of entities composing a mapping element of the alignment. More in detail, by considering the entities of each ontology numbered with integers, if the i^{th} vector cell contains the integer j, then the cell individuates the mapping element characterized by the pair of entities (e_i, e_j) where e_i is the i^{th} entity of the first ontology and e_j is the j^{th} entity of the second one. In order to improve performance of the sequential version of *MemeOptiMap*, for the parallel version, we consider the following change: if the second ontology has a cardinality minor than the first one, then the i^{th} vector cell containing the integer j corresponds to the mapping element characterized by the pair of entities (e_j, e_i) where e_j is the j^{th} entity of the first ontology and e_i is the i^{th} entity of the second one. In short, the indices of the integer vector are related to the ontology with minor cardinality, whereas, the contained integer numbers are related to the other one. Given this chromosome structure, its size is

Table 4.9: Coordinator agent behvior

Input: two ontologies O_1 and O_2 to align; PMA parameters (size of the subpopulation *pop_size*, number of subpopulations *num_pop*)).

Output: the best alignment (represented by the final best chromosome) between the ontologies O_1 and O_2.

1: *subpops* ← generatePopulations(*num_pop*, *pop_size*);
// Initialize randomly *num_pop* subpopulations of size *pop_size*
2: **sendSubPopulations**(); // Assign each subpopulation to an optimizing agent
3: **for** $i = 1 \rightarrow num\_pop$ **do**
4: *best_i* ← **wait**(); // Wait the evolution end of each optimizing agent
5: **end for**
6: **return** *best_chromosome* ← **getBestChromosome**(*best*);
// Select the best chromosome among all chromosomes received by optimizing agents

equal to L_{min}, i.e. the number of entities belonging to the ontology with a minor cardinality. A graphical representation of the new chromosome structure is given in Fig. 4.5.

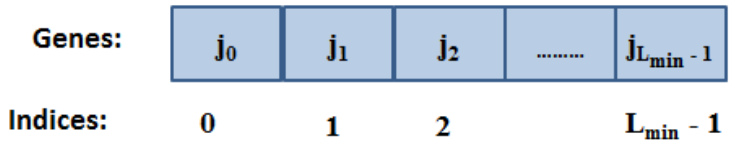


Figure 4.5: The general structure of the new chromosome used in the parallel version of *MemeOptiMap*. The chromosome is an integer vector where the indices i are equal to $0, 1, 2, \ldots, L_{min} - 1$ with L_{min} equals to the number of entities of the smaller ontology and the contained integer numbers $j_i \in \{0, 1, 2, \ldots, L_{max} - 1\}$ with L_{max} equals to the number of entities of the greater ontology.

As for features related to the local search process, each optimizing agent performs a local search process characterized by the following features:

- *Local search frequency = 1*, i.e., the local search is applied within each evolutionary cycle;
- *Order respect to genetic operators = after*, i.e., local refinement is executed after crossover and mutation operators;
- *Individual selection mechanism = a portion*, i.e., only a portion of population (the best chromosomes) is improved by the local search process;

- *Local search intensity = equal to n local search iterations*, i.e., each local search process ends after running n iterations.
- *Local search method = hill climbing*, i.e., the selected local search method is Hill Climbing search.

Table 4.10: Optimizing agent behavior

Input: two ontologies O_1 and O_2 to align; subpopulation received by coordinator agent *subpop*, MA parameters (size of the subpopulation *pop_size*, crossover rate p_c, mutation rate p_m, termination criteria t_m), local search parameters (local search termination criteria t_l, number of the best chromosomes selected for local search refinement *local_portion*), PMA parameters (migration interval *migration_interval*, number of chromosomes to be migrated *migration_rate*).

Output: the best alignment (represented by the final best chromosome) between the ontologies O_1 and O_2.

```
1: pop ← subpop);                                   // Initialize population to the subpopulation
                                                    // received by the coordinator agent
2: gen ← 0;                                         // Initialize number of iterations
3: while isFalse(t_m) do
4:    evaluateFitness(pop);                         // Evaluate fitness value for each chromosome
5:    parents ← executeSelection(pop);              // Select parents to generate new chromosomes
6:    pop ← executeCrossover(parents, p_c);         // Crossover chromosomes
                                                    // according to a crossover rate p_c
7:    pop ← executeMutation(pop, p_m); // Mutate chromosomes with a mutation probability p_m
8:    evaluateFitness(pop);                         // Evaluate fitness value for new chromosomes in pop
9:    best_chromosomes ← getBestChromosomes(pop, local_portion);   // Select the best
                                                    // chromosomes of the current subpopulation
10:   local_chromosomes ← executeLocalSearch(best_chromosomes, t_l);   // Execute
                                                    // the local search process on the best chromosomes
11:   pop ← replaceWorstChromosome(pop, local_chromosomes);   // Replace the worst
                                                    // chromosomes with that refined by the local process
12:   if mustMigration(gen, migration_interval) then
13:      best_chromosomes ← getBestChromosomes(pop, migration_rate);   // Select the best
                                                    // chromosomes to be migrated
14:      send(best_chromosomes);                    // send the best individuals
                                                    // to next neighbouring subpopulation
15:      new_chromosomes ← receive();               // receive migration_interval individuals
                                                    // from previous neighbouring subpopulation
16:      subpops_i ← replaceWorstChromosome(new_chromosomes);   // Replace the worst
                                                    // chromosomes with that migrated
17:   end if
18:   gen ← gen + 1;                                // Increment number of iterations
19: end while
20: return best_chromosome ← getBestChromosome(pop);   // Send the best chromosome
                                                    // of proper population to coordinator agent
```

It is worth noting that the chosen local search method is the Hill Climbing that, in the previous section, has been shown to be the best local method for the sequential version of *MemeOptiMap*.

Finally, it is necessary to give more details about the migration process. In general, migration procedure is a crucial step in PMAs. It is controlled by different parameters [21]:

- *Migration rate* which determines how many individuals migrate from a sub-population to the other one;
- *Migration frequency (migration interval)* which determines how often migrations occur;
- *Migration topology* which determines the destination of the migrants;
- *Migration policy* which determines which individuals migrate and which are replaced at the receiving deme.

In our architecture, each optimizing agent executes a migration after each generation. The exchange scheme of chromosomes is a one-way ring topology [128]. Each optimizing agent sends the best chromosomes and replaces the worst ones with the received migrants. As for the number of migrants, our approach follows the strategy to select a migration rate equal to the number of individuals that are improved by local search procedure.

4.4.2 Experimental results

This section is devoted to present a set of experiments performed in order to compare the performances of the parallel and sequential versions of *MemeOptiMap*. In detail, the comparison has been carried out by means of Wilcoxons signed rank test (see section 3.7.1). The experiments have involved the well-known standard benchmark dataset provided by the Ontology Alignment Evaluation Initiative (OAEI), related to the OAEI 2010 competition[1] (see section 2.3.2.1). The performance of compared systems are evaluated in terms of both quality of produced alignments and computational time. In particular, the quality of the alignments is computed by using the well-known *F-measure* (see definition 7). In our tests, in order to evaluate together the two metrics (alignment quality and computational time), we have computed the normalized ratio between them in range $[0, 1]$. This

[1] http://oaei.ontologymatching.org/2011

solution respects the desired requirement that the performances of the compared approaches result better when alignment quality increases and computational time decreases. In particular, the F-measure values and the computational times used to compute the ratio to be used as data samples for our tests are obtained by computing the average value over ten runs.

In order to perform our experiments, we have implemented parallel version of *MemeOptiMap* by considering a coordinator agent and two optimizing agents characterized by the parameter configuration shown in Table 4.11. This configuration has been chosen in an empirical way by performing preliminary experiments aimed at achieving good values for F-measure. The collaborative agents have been implemented on a single machine with a dual-core processor Intel i5 and by using Jade[1] and Ateji libraries[2]. In order to complete the description of experiments, the detail about the hardware configuration used to run the algorithms is provided:

- Processor: Intel Core i5;
- CPU Speed: 2.3 GHz;
- RAM Capacity: 4GB.

The sequential version of *MemeOptiMap* has been implemented and executed by using the same parameter configuration presented in Table 4.11. As aforementioned, the comparison between parallel version and sequential one is carried out in terms of ratio between F-measure values and computational times. These values for each test cases belonging to exploited data set are reported in Table 4.12.

As shown in Table 4.12, parallel proposal outperforms the sequential one for the 100% of benchmark test cases. However, in order to statistically verify the validity of this result, we have performed a Wilcoxon's signed rank test by considering as sample data the values presented in the Table 4.12. The Wilcoxon's test states that the parallel and collaborative approach outperforms the sequential one at 1% significance level.

[1]http://jade.tilab.com/
[2]http://www.ateji.com/

Table 4.11: Parameter Configuration

Parameter	**Value**
Population size	60 chromosomes
Subpopulation size	30 chromosomes
Crossover rate	0.8
Mutation rate	0.02
Local search intensity	50 iterations
Local search individual selection mechanism	5% of population size
Local search method	Hill Climbing with a neighborhood of 50% of population size
Termination condition	10 iterations or no fitness improvements for twice
Migration rate	10% of subpopulation size
Similarity Measures	*Entity Name Distance Matcher* *Entity Text Distance Matcher* *Individual Distance Measure* *Numbered Hierarchy Distance Measure* *Word Net Synonymy Name Distance Measure* *Super Hierarchy Distance Measure*
Aggregation	OWA operator
Weights	[0.2, 0.2, 0.15, 0.15, 0.15, 0.15]
Threshold	0.5

Table 4.12: The comparison between parallel and sequential versions of *MemeOptiMap*

Benchmark No.	***Sequential version***	***Parallel version***	***Rel. Improv.***	***Benchmark No.***	***Sequential version***	***Parallel version***	***Rel. Improv.***
101	0,078	0,123	57,7%	**238**	0,086	0,140	62,7%
103	0,077	0,123	59,3%	**239**	0,600	1,000	66,7%
104	0,079	0,124	57,5%	**240**	0,098	0,158	61,0%
201	0,072	0,114	58,4%	**241**	0,527	0,984	86,7%
202	0,047	0,068	46,7%	**246**	0,578	0,956	65,5%
203	0,123	0,171	39,4%	**247**	0,100	0,182	81,1%
204	0,076	0,121	58,9%	**248**	0,035	0,052	48,0%
205	0,074	0,118	59,1%	**249**	0,045	0,068	50,6%
206	0,071	0,112	57,7%	**250**	0,340	0,509	49,8%
207	0,077	0,123	59,0%	**251**	0,054	0,080	48,1%
208	0,118	0,165	40,3%	**252**	0,042	0,060	42,9%
209	0,089	0,128	44,7%	**253**	0,038	0,053	40,5%
210	0,087	0,125	44,2%	**254**	0,127	0,187	47,7%
221	0,093	0,149	59,3%	**257**	0,287	0,523	82,4%
222	0,093	0,149	60,6%	**258**	0,055	0,079	43,5%
223	0,089	0,143	60,8%	**259**	0,042	0,063	48,3%
224	0,080	0,126	57,6%	**260**	0,445	0,697	56,7%
225	0,079	0,128	61,7%	**261**	0,063	0,090	43,2%
228	0,376	0,634	68,8%	**262**	0,122	0,195	60,4%
230	0,090	0,147	63,2%	**265**	0,480	0,755	57,3%
231	0,076	0,123	61,3%	**266**	0,058	0,097	67,2%
232	0,093	0,151	63,0%	**301**	0,122	0,212	74,4%
233	0,569	0,927	63,0%	**302**	0,280	0,435	55,7%
236	0,452	0,686	51,6%	**303**	0,072	0,113	56,7%
237	0,089	0,145	63,5%	**304**	0,079	0,126	59,8%

Chapter 5

MemeMetaMap: A Memetic Meta-Matching for the Ontology Alignment

The goal of this thesis is to explore the application of memetic algorithms to face the ontology alignment problem. In the previous chapter, we have presented our first proposal consisting in using memetic algorithms to perform an ontology alignment process as an optimization one. In this chapter, instead, we discuss our second contribution to ontology alignment researches consisting in producing satisfactory alignments by addressing the nested ontology meta-matching problem (see section 5.1). In detail, we implement a meta-matching system, named *MemeMetaMap*, which exploits a memetic algorithm for optimizing the selection of the best ontology alignment parameters (weights and threshold). After all details about the designed system are described (see section 5.2), the chapter ends by describing one of its possible extensions based on fuzzy logic control theory implemented to overcome the *MemeMetaMap*'s dependence from some specific instance parameters (see section 5.3).

5.1 The ontology meta-matching problem

Recent trends in Information and Communication Technology (ICT), such as the cloud computing, are aimed at providing integrated services by virtualizing the knowledge spread on the web through a semantic representation of information. In these application scenarios, ontologies could represent the most appropriate technology for supporting integration and exchange of knowledge thanks to their capability of formally representing information and giving a common meaning to shared resources. However, the ability of ontologies in managing disparate information is limited by the so-called *semantic heterogeneity problem* (see section 2.2). This problem is due to the enormous variety of ways that a domain of interest can be conceptualized and, consequently, it leads to the creation of different ontologies with contradicting or overlapping parts [121]. As a consequence, it is necessary to define a so-called *ontology alignment process* whose aim is to detect a set of correspondences, named *alignment*, involving semantically related entities [107] in order to lead the different ontologies into a mutual agreement. A typical procedure applied by an ontology alignment system is to associate to all possible pairs of entities (one for each involved ontology) a confidence value, and, successively, to perform a *threshold-based filter operation* aimed at retaining only the pairs of entities with a confidence value such as to estimate it to be correct correspondences. A so-called *similarity measure* or *matcher* is used to compute the confidence value. Depending on the characteristics of ontologies under alignment, each matcher behaves more or less well in the detecting of semantic matchings among them. Therefore, the common strategy to compute a confidence value for a pair of entities is to aggregate different similarity measures through a weighted approach, where each weight represents to what degree each similarity measure should impact the alignment result [136]. Since the quality of alignment process is strongly affected by the weights used for the similarity aggregation task and the threshold exploited for the filter operation, these parameters should be opportunely chosen. The selection of the appropriate ontology alignment process parameters is known as *meta-matching problem*.

Over the years, different approaches have been investigated to find the most appropriate values for ontology alignment process parameters (see Sect. 2.4).

Mainly, they can be organized in two groups: *heuristic meta-matching systems* which exploits genetic algorithms [90] or greedy strategies [91] and *machine learning meta-matching systems* which mainly exploits neural networks [70]. However, all these methods have their drawbacks: (1) they often do not achieve high quality results in terms of alignment accuracy; or (2) they rely on inexpert users' decisions, or (3) they require rich data sets or knowledge about features of ontologies under alignment usually not available in real application scenarios.

In order to address these issues, in our research work, we design a new ontology alignment system based on a memetic meta-matching algorithm [4] , named *MemeMetaMap*, which aims at efficiently identifying both the weights for the similarity aggregation task and the similarity threshold regardless of the knowledge about the ontology features, data availability and user intervention. Since this new approach mainly deals with the optimization of ontology alignment process parameters, it can be considered as a *meta-optimization* approach, and, as a consequence, it differs from other evolutionary algorithms based works [135][16], including our first contribution presented in section 4.2, modeling the ontology alignment process as a global optimization problem. In the next section, all details about *MemeMetaMap* are given.

5.2 *MemeMetaMap* System

This section aims at presenting a new ontology alignment system based on a memetic meta-matching algorithm, named *MemeMetaMap*, capable of tuning the ontology alignment process parameters (weights and threshold) to produce high quality alignments. After a detailed description of architecture and its main components, the section ends with a discussion about the features of produced alignments in terms of matching dimensions, implementative details and some experimental results.

5.2.1 Architecture of *MemeMetaMap*

Our ontology alignment system is characterized by an architecture, presented in Fig. 5.1, which is composed of four principal components: the *pre-processing mod-*

ule, the *optimization module*, the *matcher database* and the *alignment module*. Precisely, the pre-processing module allows parsing input ontologies and extracting information necessary for performing the whole ontology alignment process. The optimization module, which represents the core of *MemeMetaMap*, performs a memetic algorithm to optimize weights and threshold necessary to, respectively, aggregate more similarity measures and to execute a filtering procedure aimed at selecting only valid correspondences. The matcher database is devoted to store similarity measures exploited by the alignment module. Finally, the alignment module is devoted to compute an alignment starting with a sub-optimal solution computed by the optimization module by using the ontological information provided by the pre-processing module and the similarity measures stored in the the matcher database.

Hereafter, a more detailed description of the main components is given.

5.2.1.1 The pre-processing Module

The pre-processing module is devoted to load and parse the two input ontologies under alignment in order to extract the ontological information useful for the whole ontology alignment process. In this research work, we consider ontologies under alignment modeled through OWL language. Therefore, the pre-processing module extracts information encoded in OWL by performing the following three steps: extraction of three distinct sets containing classes, properties and individuals; extraction of annotations such as labels and comments; extraction of axioms such as subclasses, subproperties, etc. These steps allow identifying the local context for each entity of the input ontologies containing *textual information* (name, label and comments), *structural information* (names and number of super or sub concepts), and *instance information* (names of individuals if existing). All this information allows the characterization of an entity by giving it a meaning.

5.2.1.2 The optimization module

The optimization module performs a memetic algorithm (MA) for optimizing the ontology alignment process parameters (weights and threshold) used by the alignment module to compute a sub-optimal alignment. Therefore, it is composed

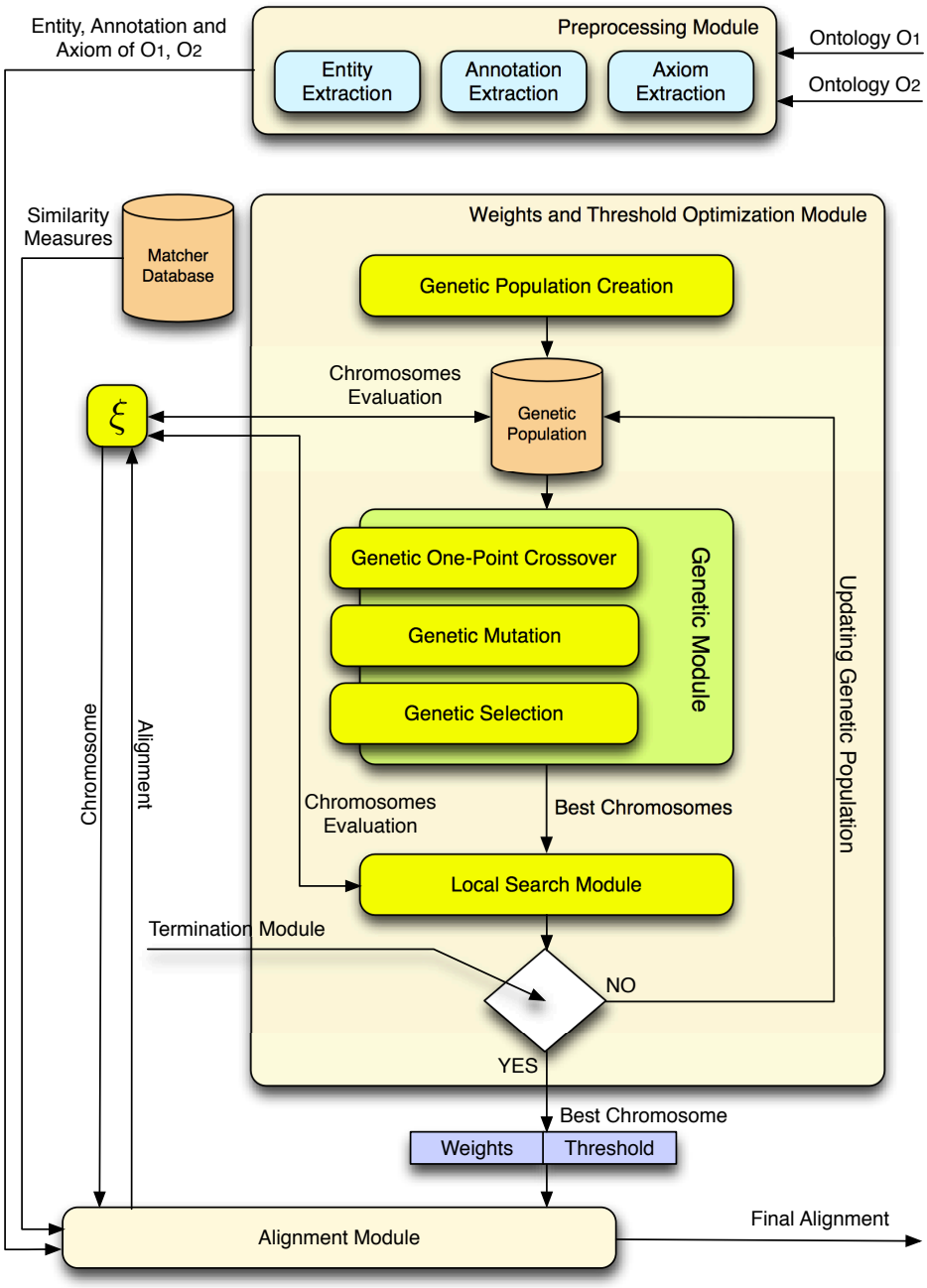


Figure 5.1: The architecture of our ontology alignment system

of a set of sub-modules aimed at implementing the general evolution of a MA.

MAs are population-based search methods which combine evolutionary algorithms (in our specific case, genetic algorithms) with local search strategies. In

fact, MAs are able to integrate local search processes within the global search successive generations in order to refine population individuals. In general, MAs try to solve an optimization problem by manipulating a *population* of potential solutions. Precisely, they operate on encoded representations of the solutions, called *chromosomes*. The algorithm evolution progresses successive *generations*. In each generation, a *selection process* provides the mechanism for selecting better solutions to survive. Each solution is evaluated by means a *fitness function* that reflects how good it is. In each generation, a recombination process of genetic material is simulated through two operators: *crossover* that exchanges portions between two randomly selected chromosomes and *mutation* that causes random alteration of the chromosome genes. Moreover, different from genetic algorithms, MAs execute a local search process within each generation. The algorithm evolution terminates when prefixed conditions such as the maximum number of generations are reached. See section 3.5 for a more detailed description of MAs.

In our approach, a MA is exploited to evolve and refine a population of chromosomes, encoding the weights and threshold values used by the ontology alignment process to build a corresponding alignment. At the end of the evolutions, the algorithm will return a suitable configuration of the ontology alignment parameters, and, consequently, a sub-optimal alignment. The whole optimization task is performed by using the following sub-modules: *genetic population creation module*, *fitness function module*, *genetic module*, *local search module* and *termination module*.

The genetic population creation module The genetic population creation module is devoted to build a random population by following the chromosome structure presented in Fig. 5.2. In detail, by considering that each chromosome represents a potential solution to problem of optimizing the ontology alignment process parameters, it contains a possible combination for the set of weights, indicating the contribution of each similarity measure, and the threshold value t, used to filter correspondences between the ontologies under alignment. Because all the considered values (weights and threshold) are numbers belonging to the interval $[0, 1]$, the chromosome can be represented as a double vector where the

first genes represent the weights and the last gene represents the threshold. Hence, by considering h similarity measures, our chromosome has a length equal to $h+1$.

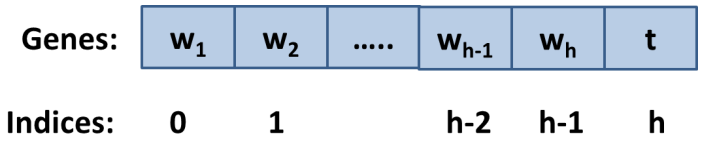


Figure 5.2: Graphical representation of a chromosome

The Fitness Module The fitness module is devoted to compute a fitness score for each chromosome of the population by using a fitness function ξ. In detail, ξ evaluates the goodness of a chromosome by estimating the quality of the corresponding alignment. In order to achieve this goal, the fitness module is supported by the alignment module which builds an alignment starting from the chromosome under evaluation (see Sect. 5.2.1.3). Different from other approaches [98][53], the alignment quality is not based on common measures such as precision, recall and F-measure, but, it takes into account the number of the correspondences belonging to the alignment and their confidence values. This choice allows our approach not to require the exploitation of a reference alignment to work, and as a consequence, it can be applied into real world scenarios. More in detail, similar to [16], the quality of an alignment is calculating by taking into account these two reasonable observations:

- the higher the average of the confidence values of the correspondences and the better the alignment quality;
- by considering the same average of the confidence values, the higher the number of correspondences and the better the alignment quality.

Hereafter, a formal definition of the fitness function ξ can be given. Let consider a chromosome σ and the corresponding alignment A, the fitness score for the chromosome σ is defined as follows:

$$\xi(\sigma) = 2 \cdot (\beta \cdot \Phi(|A|) + (1 - \beta) \cdot f(A)) \tag{5.1}$$

where $|A|$ is the number of correspondences of the alignment corresponding to the chromosome under evaluation, Φ is a function of normalization in range $[0, 1]$ and f is the function which computes the average of confidence values of the correspondences belonging to the alignment A. Precisely, f is defined as follows:

$$f(A) = \frac{\sum_{i=1}^{|A|} \eta_i}{|A|}$$

where η_i is the confidence value of the i^{th} correspondence belonging to the alignment A. In short, the function ξ is a sum weighted by β, a real value in $[0, 1]$, acting as tuning parameter useful for generating ontology alignments characterized by high precision ($\beta < 0.5$) or high recall ($\beta > 0.5$).

The Genetic Module The genetic module is devoted to perform the global search process by applying the following traditional genetic operators: *single-point crossover* and *mutation*. In general, the crossover operator takes two chromosomes called *parents* and produces two new chromosomes, called *children*, by exchanging the genes of the parents. In literature, there exist different kinds of crossover. In this work, we exploit the traditional single-point version. More in detail, the crossover operator randomly selects two chromosomes from the population and "mates" them by randomly picking a gene and then swapping that gene and all subsequent genes between the two chromosomes. The crossover operator is applied with a certain rate r_c. Therefore, it is performed $1/r_c$ as many times as there are chromosomes in the population. Instead, the mutation operator runs through the genes in each of the chromosome in the population and mutates them in statistical accordance to the given mutation rate p_m. Therefore, the number of mutation operations is not deterministic.

The application of these genetic operators produce a set of new chromosomes. They are added to the population and evaluated by means of the fitness module. After these conventional genetic steps, our algorithm applies a genetic selection operator to generate the new population whose the best chromosomes will be involved in the local search procedure. Compliant with conventional genetic algorithm design, our approach uses as selection operator the *roulette wheel selection* method. It consists in giving to each chromosome a probability of being selected

which is directly proportionate to its fitness score. In detail, if s_i is the fitness score of i^{th} individual of the population, its probability p_i of being selected is $p_i = \frac{s_i}{\sum_{k=1}^{N} s_k}$, where N is the number of individuals in the population. Therefore, the best solutions will have more possibilities of belonging to the next generated population.

The Local Search Module The local search module is devoted to perform a local search process in order to refine the population deriving from the application of the genetic operators. Many issues must be addressed about the integrated local search process in order to design an efficient MA as described in section 3.5.

Our local search module is characterized by the following features:

- *Local search frequency*: Our local search module performs a local refinement within each evolutionary cycle;
- *Individual selection mechanism*: Our local search module improves only a portion of the population composed of the best chromosomes;
- *Local search intensity*: Our local search module executes a local search process which takes n iterations;
- *Local search method*: Our local search module performs the stochastic version of the Hill Climbing search (see section 3.3).

5.2.1.3 The alignment module

The alignment module, whose a more detailed architecture is shown in Fig. 5.2.1.3, is devoted to build an alignment starting with a chromosome by using the ontological information extracted by the pre-processing module and the similarity measures stored in the matcher database. It performs its task both to support the fitness module and to produce in output the final alignment starting from the optimal solution (the best chromosome of the population after the achievement of termination criteria) computed of designed memetic algorithm.

In detail, the alignment module computes the following steps:

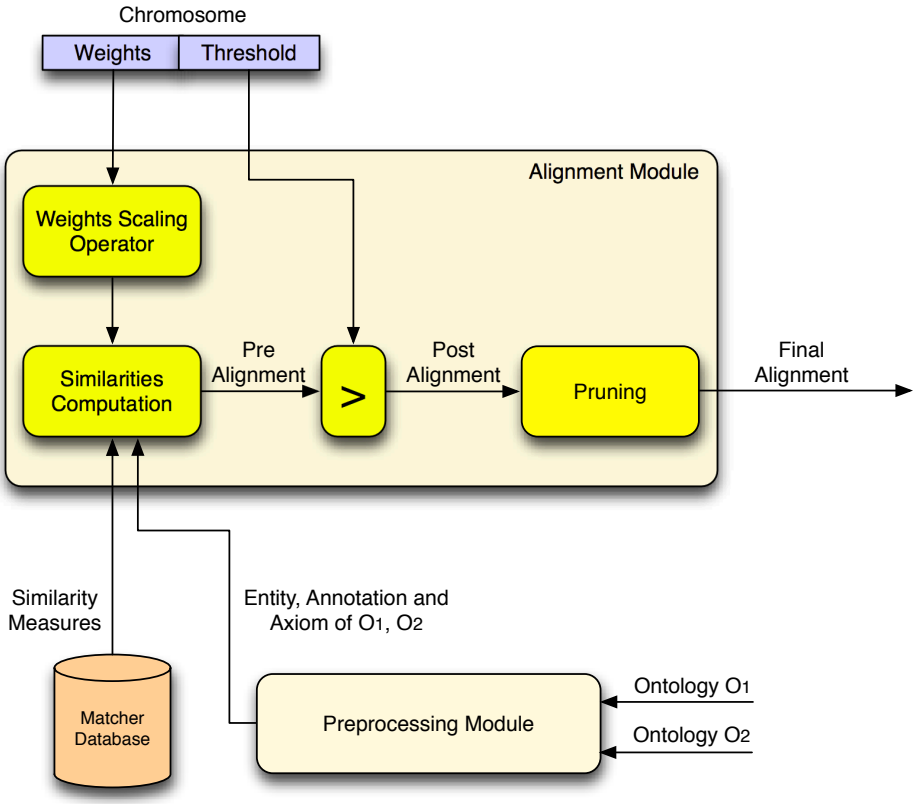


Figure 5.3: The architecture of the Alignment Module

- scaling of the weights by following the equation 5.2. Indeed, as defined in 2.1, the sum of all weights has to be equal to 1. Therefore, in order to obtain the real value of weights to be used in the similarity aggregation task, it will be necessary a scaling procedure. In detail, each one of the first h values of the chromosome (representing weights) has to be scaled conforming to the sum of all first h values. Formally:

$$w_i = \frac{g_i}{\sum_{i=1}^{h} g_i} \tag{5.2}$$

  where g_i is the i^{th} value of the chromosome and w_i is the i^{th} weight will be used to perform the similarity aggregation task;

- building of a pre-alignment by setting for each pair of entities between the two ontologies to be aligned a confidence value calculated as the aggregated

similarity $sim_{aggregate}$ defined in equation 2.1. In detail, the similarity aggregation task involves the scaled weights and the similarity measures present in the matcher database;

- building of a post-alignment by filtering the pair of entities with a confidence value greater or equal to the threshold value (the last gene of the chromosome);

- building of a final alignment by performing a pruning procedure which consists in selecting only a correspondence for each entity of the so-called *goal ontology*. The selected correspondence has the highest confidence value. If there are more correspondences with the highest confidence value, the procedure executes a random choice. The goal ontology is represented by the ontology with the minor number of entities.

By analyzing the performed steps, the alignment module produces alignments injective with regard to no-goal ontology, i.e., all the entities of the goal ontology are part of at most one correspondence of the produced alignment. In other words, by using a usual notation, the produced alignments are characterized by a multiplicity $? : *$ or $* :?$, respectively, depending on whether the goal ontology is the first one or the second one.

However, a more detailed discussion on matching dimensions is given in the next section.

5.2.2 Discussions on Matching dimensions

Beside the general scheme presented in Definition 2, it is useful to consider a collection of additional features related to the ontology alignment process, known as *dimensions* (see section 2.3.1.1). They represent constraints or restrictions on the ontology alignment process which influence the behavior of ontology alignment systems and, as a consequence, they can be used for performing their classification [43]. This section is devoted to present the dimensions and the corresponding values which characterize the behavior of *MemeMetaMap*. This description allows to highlight the features of the produced alignments, and, at the same time, categorize *MemeMetaMap* in the state of the art.

In detail, the dimensions and the corresponding values which characterize *MemeMetaMap* are:

- *heterogeneity of input ontologies*: the input ontologies must be coded by the same language since *MemeMetaMap* does not provide a mechanism for converting them in a common language;
- *language of input ontologies*: *MemeMetaMap* supports RDF and OWL models;
- *complete/update of input alignment*: *MemeMetaMap* computes a whole ontology alignment from scratch. Therefore, our algorithm does not take in input an initial partial alignment or in other words, the initial alignment is equal to $\emptyset$;
- *Resources*: *MemeMetaMap* exploits WordNet as dictionary for the linguistic similarity computation;
- *Proper parameters*: *MemeMetaMap* needs a collection of parameters for tuning the behavior of the memetic algorithm performed by the memetic module;
- *multiplicity of output alignment*: *MemeMetaMap* produces alignments with multiplicity $? : *$ or $* :?$, respectively, related to cases: cardinality of the first ontology lesser than cardinality of the second one and vice versa;
- *relations of output alignment*: currently, *MemeMetaMap* takes into account only equivalence relations.

Moreover, as for the high level classification in schema or instance-based approach [43], *MemeMetaMap* is hybrid since it can exploit similarity measures considering both schema and instance-based information.

5.2.3 Implementative details

MemeMetaMap has been implemented in Java[1]. It is mainly based on two Java libraries:

[1] http://www.java.com/en/

Table 5.1: Parameters of the designed Memetic Algorithm for meta-matching problem

Kind	Name	Description
Genetic parameters	population	the number of chromosomes
	crossoverRate	the probability of crossover operator
	mutationRate	the problability of mutation operator
	termination	the termination criteria to be chosen among number of iterations, number of fitness evaluations, convergence, precision, recall, F-measure
Local search parameters	intensity	the number of iterations performed by local search
	Local search selection mechanism	which and how many chromosomes are selected for local refinement
	method	the local search method
Ontology alignment parameters	matchers	the used set of similarities measures
	aggregation	the used aggregation strategy

- Alignment API[1] which serves as an interface to ontologies and alignments;
- JGap[2] used to implement genetic components of designed memetic algorithm such as chromosomes, genetic operations and so on.

As for the local search methods, we have designed a custom implementation. The algorithm setting can be configured by a parameter file. The complete list of parameters is reported in table 5.1.

Once all details about *MemeMetaMap* have been described, in the following section, some experimental tests show the suitability of our approach and its high performances.

5.2.4 Experimental results

This section presents a set of experimental results aimed at showing the suitability of *MemeMetaMap* to address the ontology alignment problem. The whole experimental session involves the standard benchmark track[3] provided by OAEI

[1] http://alignapi.gforge.inria.fr/
[2] http://jgap.sourceforge.net/
[3] http://oaei.ontologymatching.org/2010/

(see section 2.3.2.1), i.e., a set of test cases built around a reference ontology which is dedicated to model the domain of bibliography and many variations of it [73]. According to OAEI policies, our system computes alignments containing only correspondences, respectively, between ontology classes and properties, and excluding individual matching. The performances yielded by *MemeMetaMap* are assessed by means of standard evaluation measures considered by OAEI: *precision* (see definition 5), *recall* (see definition 6) and *F-measure* (see definition 7). Table 6.2 shows the configuration of *MemeMetaMap* used in the experiments. It represents a trade-off setting obtained in empirical way. Instead, Table 5.3 shows the results of the experiments in terms of precision, recall and F-measure with respect to the given reference alignment. In particular, the reported values represent the average on ten runs.

Table 5.2: Memetic Meta-matching Configuration

Parameter	**Value**
Population size	30 chromosomes
Crossover rate	0.9
Mutation rate	0.02
Maximum number of local iterations	20
Local search selection mechanism	15% of population
Termination condition	250 fitness evaluations
β	0.2
Similarity Measures	*Entity Name Distance Matcher* *Entity Text Distance Matcher* *Word Net Synonymy Name Distance Measure* *Super Hierarchy Distance Measure* *Numbered Hierarchy Distance Measure* *Individual Distance Measure*
aggregation strategy	OWA operator

As shown in the Table 5.3, *MemeMetaMap* achieves the best performances in the first and third test case ranges (# 101-104 and # 221-247). Besides, it yields good results also as for the second test case range (# 201-210), except for the test case #202 which presents some difficulties regarding the use of random labels and the abolition of an important feature such as comments in the involved

Table 5.3: Experimental results for the standard benchmark track

No.	***Precision***	***Recall***	***F-measure***	***No.***	***Precision***	***Recall***	***F-measure***
101	1	1	1,00	**238**	1	1	1,00
103	1	1	1,00	**239**	0,97	1	0,98
104	1	1	1,00	**240**	0,81	0,94	0,87
201	1	0,95	0,97	**241**	1	1	1,00
202	0,59	0,29	0,39	**246**	0,97	1	0,98
203	1	1	1,00	**247**	0,88	0,99	0,93
204	1	1	1,00	**248**	0,69	0,15	0,25
205	1	0,98	0,99	**249**	0,53	0,33	0,41
206	0,98	0,94	0,96	**250**	0,5	0,5	0,50
207	0,99	0,94	0,96	**251**	0,62	0,22	0,32
208	0,98	0,98	0,98	**252**	0,45	0,25	0,32
209	0,7	0,69	0,69	**253**	0,43	0,24	0,31
210	0,76	0,75	0,75	**254**	0,14	0,14	0,14
221	1	1	1,00	**257**	0,5	0,5	0,50
222	1	1	1,00	**258**	0,54	0,24	0,33
223	1	1	1,00	**259**	0,3	0,3	0,30
224	1	1	1,00	**260**	0,31	0,31	0,31
225	1	1	1,00	**261**	0,16	0,32	0,21
228	1	1	1,00	**262**	0,12	0,12	0,12
230	0,94	1	0,97	**265**	0,24	0,25	0,24
231	1	1	1,00	**266**	0,16	0,32	0,21
232	1	1	1,00	**301**	0,94	0,81	0,87
233	1	1	1,00	**302**	0,7	0,54	0,61
236	1	1	1,00	**303**	0,61	0,78	0,68
237	1	1	1,00	**304**	0,88	0,95	0,91

varied ontology. As for the real test cases (# 301-304), the results are good but not optimal due to the cardinality of the reference alignment (it is not ?:* or *:? for all test cases) or to kind of involved matchings including also subsumption relations. The worst results are obtained in the fourth test case range (# 248-266) because of the decreasing number of features available in the ontologies.

5.3 A fuzzy extension for *MemeMetaMap*

As described in the previous section, *MemeMetaMap* is not capable of producing alignments with the same high quality on all alignment task instances. This weakness is mainly due to dependence of its behavior on a set of specific instance parameters affecting the behaviour of the designed fitness function (definition 5.1). In order to overcome this dependence, fuzzy logic theory and its most common application, i.e., Fuzzy Logic Controllers (FLCs), seem to be suitable by means of allowing to adapt *MemeMetaMap*'s parameters to each specific alignment task instance. In detail, our idea is to exploit a Mamdani FLC [88] capable of analyzing the characteristics of the ontologies involved in a specific alignment task in order to adaptively regulate the specific instance fitness parameters, and as a consequence, improve the quality of the produced alignments. In this section, we present the work done in this research area.

5.3.1 The issue of specific instance fitness parameters

MemeMetaMap uses a memetic algorithm characterized by a fitness function (definition 5.1) whose behaviour strongly depends on the value of parameter β. As described, this parameter is useful for generating ontology alignments characterized by high precision ($\beta < 0.5$) or high recall ($\beta > 0.5$). However, since typically the *desideratum* is to produce alignments with a good trade-off between precision and recall, i.e., with a high F-measure values, β must be chosen in an opportune way in order to achieve this goal. Unfortunately, the most opportune value for β capable of reaching high F-measure values is different from alignment task to another one. In other words, the parameter β is a specific instance parameter whose value depends on features of ontologies composing the alignment task at hand. Moreover, the designed fitness function implicitly tries to maximize the number of correspondences until to achieving the number of correspondences composing the ideal optimal alignment that we denote with Θ. However, since Θ is not known during the execution of a real ontology alignment process, currently *MemeMetaMap* tries to maximize until to achieving the number of entities contained in the smaller ontology to be aligned, as described in 5.2.1.3 and reflected from cardinality of possible produced alignments. Unfortunately, this approxi-

mation of Θ might be very far from the truth. Therefore, the performance of *MemeMetaMap* could be improved by a better approximation technique. Let formulate the value Θ for an alignment task composed of ontologies O_1 and O_2 as follows:

$$\Theta = \Omega * min(|O_1|, |O_2|) \tag{5.3}$$

where $|O_1|$ and $|O_2|$ represent, respectively, the cardinality of ontologies O_1 and O_2, *min* is a function which computes minimum between two values and $\Omega \in [0, 1]$ is a tuning parameter which allows better approximating the value for Θ. By using this new idea to approximate the value for Θ, the fitness function ξ for a chromosome σ in the meta-matching algorithm must be changed as follows:

$$\xi(\sigma) = \beta \cdot (1 - \Phi(abs(\Theta - |A|))) + (1 - \beta) \cdot f(A) \tag{5.4}$$

where Θ represents the number of correspondences composing the optimal alignment, $|A|$ is the number of correspondences composing the alignment under evaluation, Φ is a function of normalization in range $[0, 1]$, *abs* is a function which computes the absolute value, f is the function which computes the average of distances characterizing the correspondences belonging to the alignment A, β is a specific instance parameter used to balance between the two addends composing the fitness function. The new fitness function continues to have to be maximized.

Obviously, the formulation of this new method of approximation for the value Θ introduces a new issue: which is the optimal value for the parameter Ω in equation 5.3?. This value, like β, strongly depends on features characterizing the ontologies under alignment.

Starting from these considerations, the next section presents an extension of *MemeMetaMap* which consists in regulating the parameters (β and Ω) influencing the behavior of the designed fitness function (definition 5.4) through a Mamdani fuzzy logic controller taking into account characteristics of the specific ontology alignment problem instance.

5.3.2 A fuzzy logic controller for adapting *MemeMetaMap*

Fuzzy control theory can be considered as the most active and fruitful research area in the application of fuzzy logic. Its realization, i.e., a Fuzzy Logic Controller (FLC), is an adequate methodology for capturing and managing the approximate, inexact nature of the real world. From this point of view, FLCs let controller designers to describe complex systems using their knowledge and experience by means of linguistic IF-THEN rules differently from others controller design methodologies such as proportional-integral-derivative (PID) controllers using complex math equations. In general, the high-level structure of a FLC is shown in Fig. 5.4 [8]. The main components of a fuzzy controller are:

- fuzzy knowledge base;
- fuzzy rule base;
- inference engine;
- fuzzification subsystem;
- defuzzification subsystem.

In detail, the fuzzy knowledge base manipulates the variables used in the controlled system (such as temperature, pressure, etc.), corresponding to the knowledge used by human experts. The Fuzzy Rule Base represents the set of relations enclosed in rules between input fuzzy variables and output ones defined in the controlled system. More in detail, each rule has an if-then format, and formally the if-side is called the *antecedent part* and the then-side is called the *consequent part*. It is worth noting that the nature of consequent part of fuzzy rules permits to define two kinds of fuzzy controller: the Mamdani controller and the Takagi-Sugeno-Kang (TSK) controller [125]. The Mamdani controller uses a fuzzy set to model the consequent part of rule, whereas, the TSK controller uses the linear function of input variables. The Inference Engine is the fuzzy controller component able to extract new knowledge from fuzzy knowledge base and fuzzy rule base. Moreover, since the controlled system works with real numbers, whereas the fuzzy controller system works with fuzzy concepts, the last two subsystems,

the fuzzification subsystem and the defuzzification subsystem, are necessary to bridge this gap. More in detail, the former permits to transform the real numbers used by controlled systems into a fuzzy set used by fuzzy controller. The latter transforms the fuzzy set generated by fuzzy controller into real numbers usable by controlled system.

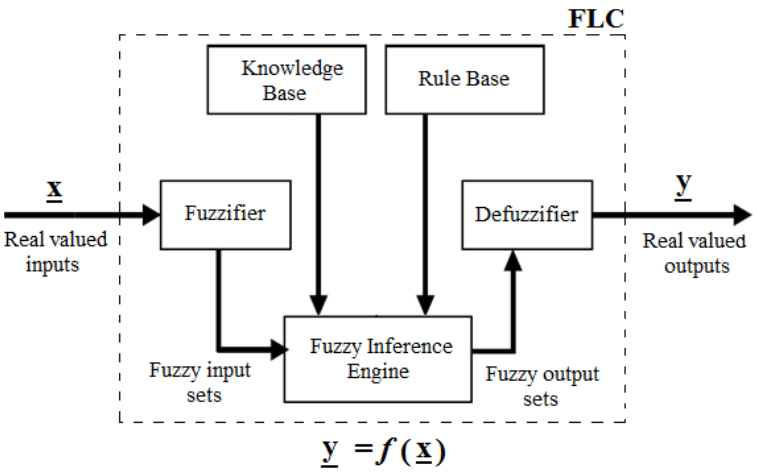


Figure 5.4: The general structure of a fuzzy logic controller

In our case, a FLC allows computing values for the parameters β and Ω influencing the behavior of the *MemeMetaMap*'s fitness function (definition 5.4) by managing the uncertainty related to features of ontologies under alignment. In detail, we design a Mamdani FLC composed of two output variables, named *Beta* and *Omega*, and four input variables, named *diff*, *sl*, *sw*, *ss*, representing factors affecting output values. In detail, *diff* represents the relative difference in percentage between the number of entities composing the two ontologies under alignment; *sl* represents how much the ontologies to be aligned are identical under a lexical point of view; *sw* represents how much the ontologies to be aligned are identical under a linguistic point of view; *ss* represents how much the ontologies to be aligned are identical under a structural point of view. Figs. 5.5-5.8 show fuzzy sets of input variables, whereas, Figs. 5.9 and 5.10 depict fuzzy sets of output ones. As you can see from Figs. 5.5-5.10, input variables are modeled with trapezoidal fuzzy sets, whereas, output ones are described with triangular fuzzy sets. As for rule base, it is composed of 81 rules. The designed inference

engine computes the conventional Mamdani's inference method [89] consisting of *min-max* operations. As for the process of defuzzification, the designed FLC uses the mean of maxima method [83] which takes the mean of the points with the strongest possibility, i.e. maximal membership. This method disregards the shape of the fuzzy set, but the computational complexity is relatively good [74].

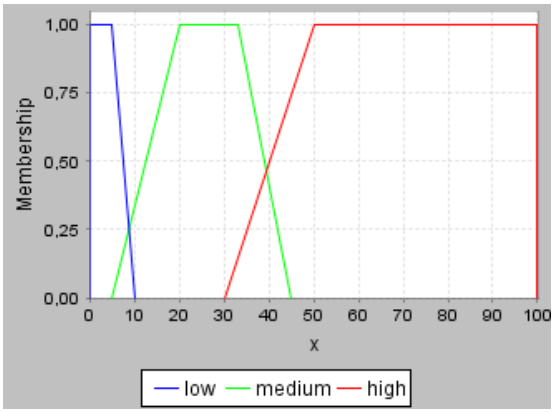


Figure 5.5: Variable *diff*

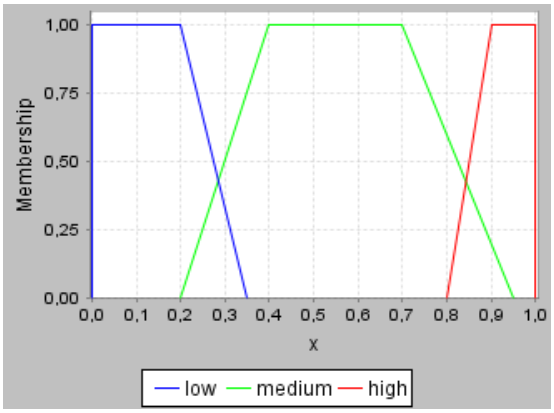


Figure 5.6: Variable *sl*

The designed FLC has been implemented in an XML-based language for modeling fuzzy controllers, named *Fuzzy Markup Language* [5][6], thanks to support of a Visual Tool [9]. Then, FML code has been integrated in the presented memetic ontology alignment system. A portion of the rulebase modeled in FML is depicted in listing 5.1.

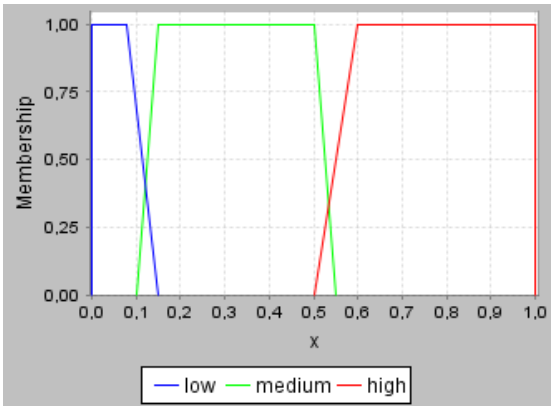


Figure 5.7: Variable *sw*

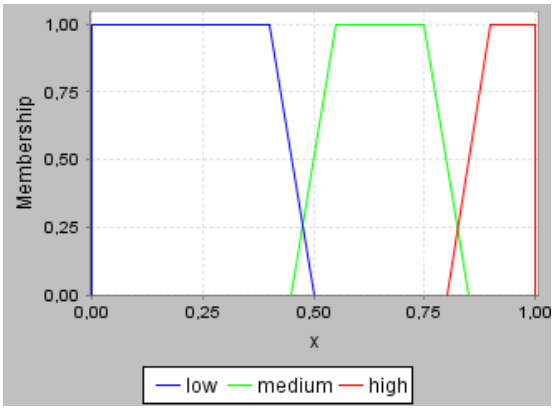


Figure 5.8: Variable *ss*

Inputs necessary to the designed FLC to compute output values for the parameters β and Ω related to a specific alignment task are calculated starting from input ontologies O_1 and O_2 under alignment as follows:

$$diff = \frac{abs(|O_1| - |O_2|)}{max(|O_1|, |O_2|)} * 100$$

$$sl = \frac{\#iden\_ent\_label}{min(|O_1|, |O_2|)}$$

$$sw = \frac{\#syn\_ent\_label}{|O_1 \times O_2|}$$

$$ss = \frac{\#sub\_ent}{|O_1 \times O_2|}$$

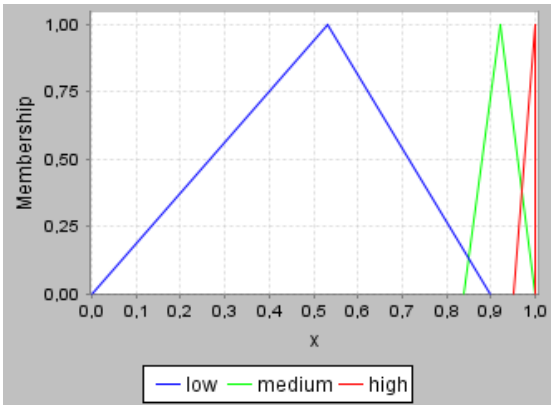


Figure 5.9: Variable *Omega*

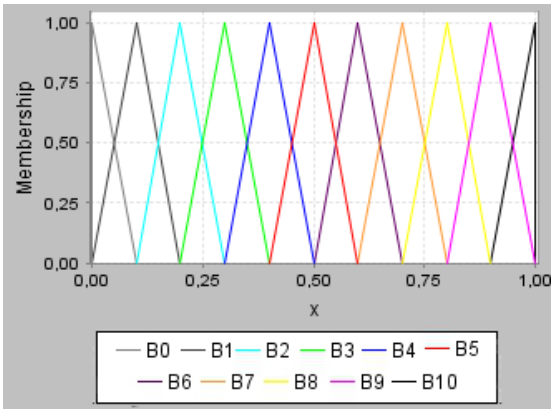


Figure 5.10: Variable *Beta*

where $|O_1|$ and $|O_2|$ represent, respectively, the cardinality (number of entities) of the two ontologies under alignment, $|O_1 \times O_2|$ represents the number of pairs composing the Cartesian product between the two ontologies under alignment, *abs* is a function which computes the absolute value, $\#iden\_ent\_label$ represents the number of identical name pairs in the entity's names of two ontologies under alignment, $\#syn\_ent\_label$ represents the number of name pairs in the entity's names of two ontologies under alignment which are synonymous; $\#sub\_ent$ represents the number of pairs between the two ontologies under alignment characterized by entities with the same number of super- and subentities. Fig. 5.11 shows output values for β and Ω starting with the following input values: $diff = 40$, $sl = 0,26$, $sw = 0,43$, $ss = 0,09$.

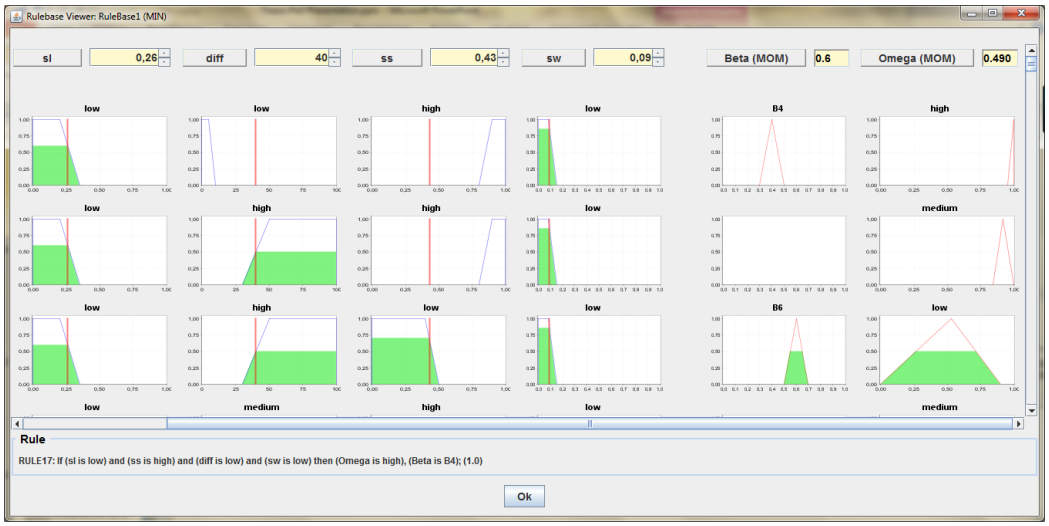


Figure 5.11: A simulation of fuzzy logic controller behavior

Finally, the general behaviour of the designed FLC is highlighted by control surfaces presented in Fig. 5.12.

5.3.3 Experimental results

This section is devoted to show how *MemeMetaMap* enhanced with the designed fuzzy logic controller yields better performance. The experiments involve the alignment tasks belonging to the well-known OAEI dataset named *standard benchmark track* (see 2.3.2.1). The improvement provided by the designed fuzzy extension has been investigated in terms of quality of produced alignments by using a well-known conformance measure, called *F-measure* (see definition 7). The comparison is carried out with the original version of *MemeMetaMap* which does not involve the fuzzy adaptation achieved through the designed fuzzy logic controller. It executes the Wilcoxons signed rank test (see section 3.7.1).

The configuration of parameters used for running *MemeMetaMap* in both versions is the following one:

- population size = 30 chromosomes;
- crossover rate = 0.9;
- mutation rate = 0.02;

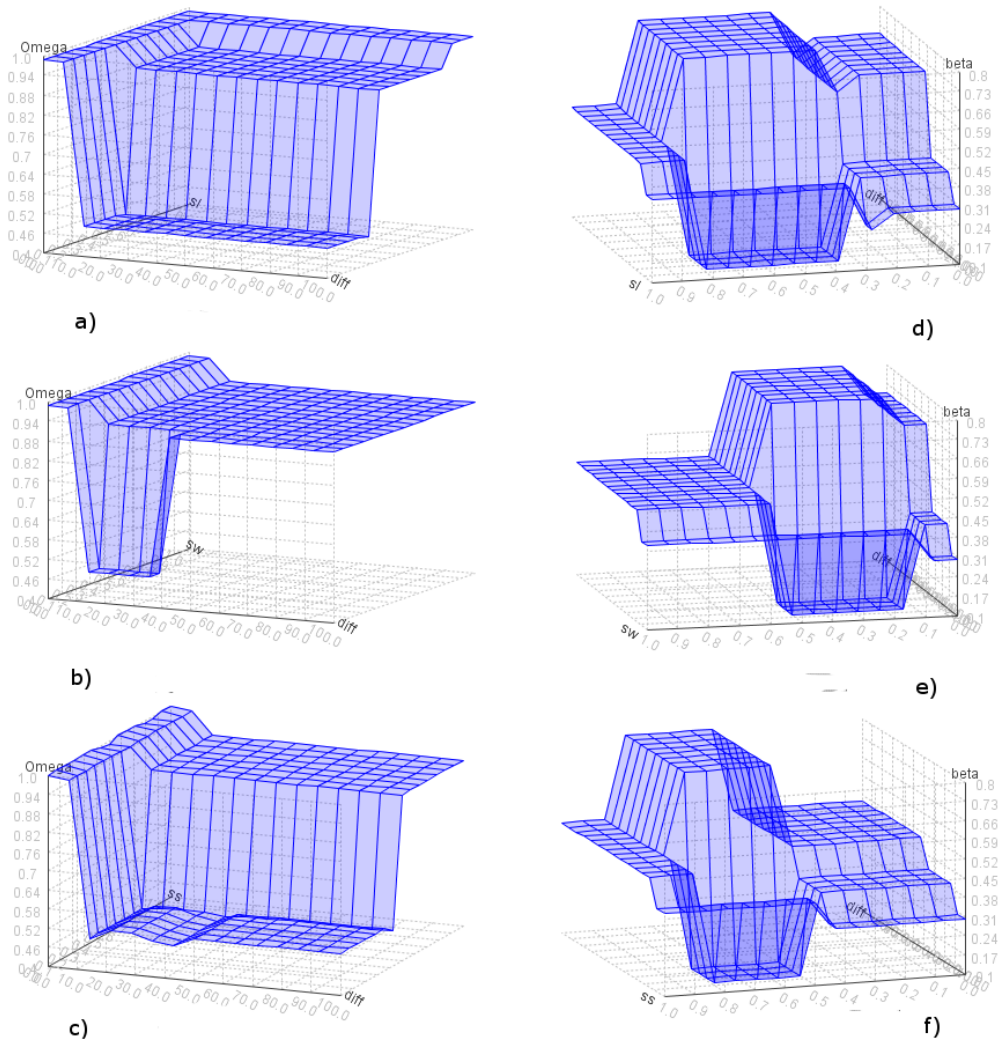


Figure 5.12: Control surfaces with variable Ω and β on axis z, respectively, in a)-b)-c) and d)-e)-f), and variables a)-d) *diff* and *sl*, b)-e) *diff* and *ss*, c)-f) *diff* and *sw*, on axis x and y

- maximum number of local iterations = 70;
- termination condition = 250 fitness evaluations;
- similarity measures = *Entity Name Distance Measure*, *Comment Distance Measure*, *Hierarchy Distance Measure*, *Domain and Range Restrictions Distance Measure* and *Word Net Synonymy Name Distance Measure*.

Table 5.4 shows F-measure values obtained by computing the average value

over fifteen runs for both compared ontology alignment systems.

Table 5.4: The comparison between the original *MemeMetaMap* and its fuzzy extension.

Benchmark No.	***Fuzzy version***	***Original version***	***Rel. Improv.***	***Benchmark No.***	***Fuzzy version***	***Original version***	***Rel. Improv.***
101	1	1	0%	**238**	1	1	0%
103	1	1	0%	**239**	0,98	0,98	0%
104	1	1	0%	**240**	0,96	0,87	10,34%
201	0,95	0,97	-2,06%	**241**	1	1	0%
202	0,44	0,39	12,82%	**246**	0,98	0,98	0%
203	1	1	0%	**247**	0,96	0,93	3,23%
204	1	1	0%	**248**	0,35	0,25	40,00%
205	0,99	0,99	0%	**249**	0,44	0,41	7,32%
206	0,96	0,96	0%	**250**	0,48	0,5	-4,00%
207	0,96	0,96	0%	**251**	0,36	0,32	12,50%
208	0,98	0,98	0%	**252**	0,32	0,32	0%
209	0,68	0,69	-1,45%	**253**	0,34	0,31	9,68%
210	0,76	0,75	1,33%	**254**	0,3	0,14	114,29%
221	1	1	0%	**257**	0,43	0,5	-14,00%
222	1	1	0%	**258**	0,36	0,33	9,09%
223	1	1	0%	**259**	0,28	0,3	-6,67%
224	1	1	0%	**260**	0,5	0,31	61,29%
225	1	1	0%	**261**	0,33	0,21	57,14%
228	1	1	0%	**262**	0,27	0,12	125,00%
230	0,97	0,97	0%	**265**	0,47	0,24	95,83%
231	1	1	0%	**266**	0,3	0,21	42,86%
232	1	1	0%	**301**	0,87	0,87	0%
233	1	1	0%	**302**	0,64	0,61	4,92%
236	1	1	0%	**303**	0,79	0,68	16,18%
237	1	1	0%	**304**	0,93	0,91	2,20%

As shown in Table 5.4, the fuzzy enhanced version of *MemeMetaMap* outperforms the original version for the 90% of test cases. However, in order to statistically verify the validity of the designed fuzzy extension, we have performed a Wilcoxon's signed rank test by considering as sample data the F-measure values presented in the Table 5.4. The Wilcoxon's test states that the fuzzy enhanced version of *MemeMetaMap* outperforms original system at 1% significance level.

Listing 5.1: FML code to model a portion of the rule base

```
<?xml version="1.0" encoding="UTF-8"?>
<FuzzyController name="FLC" ip="127.0.0.1">
....
    <RuleBase name="RB1" andMethod="MIN" orMethod="MAX"
                    activationMethod="MIN" type="mamdani">
        <Rule name="RULE1" connector="and" operator="MIN" weight="1.0">
            <Antecedent>
                <Clause>
                    <Variable>sl</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>ss</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>diff</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>sw</Variable>
                    <Term>medium</Term>
                </Clause>
            </Antecedent>
            <Consequent>
                <Clause>
                    <Variable>Omega</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>Beta</Variable>
                    <Term>B1</Term>
                </Clause>
            </Consequent>
        </Rule>
        <Rule name="RULE2" connector="and" operator="MIN" weight="1.0">
            <Antecedent>
                <Clause>
                    <Variable>sl</Variable>
                    <Term>low</Term>
                </Clause>
                <Clause>
                    <Variable>ss</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>diff</Variable>
                    <Term>medium</Term>
                </Clause>
                <Clause>
                    <Variable>sw</Variable>
                    <Term>low</Term>
                </Clause>
            </Antecedent>
            <Consequent>
                <Clause>
                    <Variable>Omega</Variable>
                    <Term>low</Term>
                </Clause>
                <Clause>
                    <Variable>Beta</Variable>
                    <Term>B4</Term>
                </Clause>
            </Consequent>
        </Rule>
        .....
    </RuleBase>
</FuzzyController>
```

Chapter 6

Evaluation: Memetic Approaches vs the State of the art

In the previous chapters, we described the development of two new ontology alignment systems based on memetic algorithms, *MemeOptiMap* and *MemeMetaMap*, and, their performances by executing some experiments. The aim of this chapter is to present, firstly, the comparison between *MemeOptiMap* and *MemeMetaMap* (see section 6.1), and then, between these memetic approaches and the existing methods in literature (see section 6.2). As shown by the led statistical tests, the exploitation of memetic algorithms results effective to face the ontology alignment problem.

6.1 Comparison between *MemeOptiMap* and *MemeMetaMap*

This section is devoted to present the statistical comparison executed between the two proposed ontology alignment systems based on memetic algorithms, named *MemeOptiMap* (see chapter 4) and *MemeMetaMap* (see chapter 5). The performances of the two ontology alignment systems are investigated both in terms of alignment quality and computational cost. The compared systems, *MemeOptiMap* and *MemeMetaMap*, are executed with configurations reported, respectively, in Tables 6.1 and 6.2. They represent a trade-off setting obtained in

empirical way which allow both systems to achieve the highest alignment quality average on all test cases of exploited dataset.

Table 6.1: Configuration for *MemeOptiMap*

Parameter	**Value**
Population size	30 chromosomes
Crossover rate	0.8
Mutation rate	0.02
Local search intensity	50 iterations
Local search individual selection mechanism	6% of population
Local search method	Hill Climbing with a neighborhood of 30 chromosomes
Termination condition	20 iterations or no fitness improvements for twice
matchers	*Entity Name Distance Matcher* *Entity Text Distance Matcher* *Word Net Synonymy Name Distance Measure* *Super Hierarchy Distance Measure* *Numbered Hierarchy Distance Measure* *Individual Distance Measure*
aggregation	OWA operator
weights	[0.2, 0.15, 0.15, 0.2, 0.15,0.15]
threshold	0.5

Hereafter, the details about the carried out comparison in terms of alignment quality and computational effort.

6.1.1 Alignment quality comparison

The first test intends to compare *MemeOptiMap* and *MemeMetaMap* in terms of quality of produced alignments. In order to achieve this aim, we use the conformance measure F-measure described in section 2.3.2.2. Table 6.3 shows the comparison. The reported values represent the average F-measure on ten runs.

As shown in Table 6.3, *MemeMetaMap* outperforms *MemeOptiMap* for the 86% of test cases. However, in order to statistically verify the validity of these

Table 6.2: Configuration for *MemeMetaMap*

Parameter	Value
Population size	30 chromosomes
Crossover rate	0.9
Mutation rate	0.02
Local search iterations	20
Local search individual selection mechanism	15% of population
Termination condition	250 fitness evaluations
β	0.2
matchers	*Entity Name Distance Matcher* *Entity Text Distance Matcher* *Word Net Synonymy Name Distance Measure* *Super Hierarchy Distance Measure* *Numbered Hierarchy Distance Measure* *Individual Distance Measure*
aggregation	OWA operator

results, we have performed a Wilcoxon's signed rank test by considering as sample data the F-measure values presented in the table 6.3. The Wilcoxon's test states that *MemeMetaMap* outperforms *MemeOptiMap* at 1% significance level.

6.1.2 Computational cost comparison

The second test intends to compare *MemeOptiMap* and *MemeMetaMap* in terms of computational effort. In order to achieve this aim, we use the performance measure, named speed, described in section 2.3.2.2. Table 6.4 shows the comparison. The reported values represent the average speed (measured in seconds) on ten runs.

As shown in Table 6.4, *MemeMetaMap* outperforms *MemeOptiMap* for the 100% of test cases. However, in order to statistically verify the validity of these results, we have performed a Wilcoxon's signed rank test by considering as sample data the F-measure values presented in the table 6.3. The Wilcoxon's test states that *MemeMetaMap* outperforms *MemeOptiMap* at 1% significance level.

However, we think that relevant improvements for both systems in terms of

Table 6.3: Comparison between *MemeMetaMap* (MMM) and *MemeOptiMap* (MOM) in terms of F-measure

No.	***MMM***	***MOM***	***Rel. Improv.***	***No.***	***MMM***	***MOM***	***Rel. Improv.***
101	1	0,99	1,01%	**238**	1	1	0,00
103	1	1	0%	**239**	0,98	0,98	0%
104	1	1	0%	**240**	0,96	0,77	24,68%
201	0,95	0,97	-2,06%	**241**	1	1	0%
202	0,44	0,35	25,71%	**246**	0,98	0,98	0%
203	1	1	0%	**247**	0,96	0,76	26,32%
204	1	0,99	1,01%	**248**	0,35	0,25	40,00%
205	0,99	0,91	8,79%	**249**	0,44	0,35	25,71%
206	0,96	0,96	0%	**250**	0,48	0,42	14,29%
207	0,96	0,97	-1,03%	**251**	0,36	0,38	-5,26%
208	0,98	0,98	0%	**252**	0,32	0,3	6,67%
209	0,68	0,64	6,25%	**253**	0,34	0,27	25,93%
210	0,76	0,72	5,56%	**254**	0,3	0,16	87,50%
221	1	1	0%	**257**	0,43	0,42	2,38%
222	1	1	0%	**258**	0,36	0,38	-5,26%
223	1	1	0%	**259**	0,28	0,3	-6,67%
224	1	1	0%	**260**	0,5	0,49	2,04%
225	1	1	0%	**261**	0,33	0,31	6,45%
228	1	1	0%	**262**	0,27	0,16	68,75%
230	0,97	0,96	1,04%	**265**	0,47	0,49	-4,08%
231	1	0,99	1,01%	**266**	0,3	0,31	-3,23%
232	1	1	0%	**301**	0,87	0,83	4,82%
233	1	1	0%	**302**	0,64	0,64	0%
236	1	1	0%	**303**	0,79	0,68	16,18%
237	1	1	0%	**304**	0,93	0,89	4,49%

computational cost could be still obtained by stressing and exploring parallel approaches.

Table 6.4: Comparison between *MemeMetaMap* (MMM) and *MemeOptiMap* (MOM) in terms of speed (measured in seconds)

No.	***MMM***	***MOM***	***Rel. Improv.***	***No.***	***MMM***	***MOM***	***Rel. Improv.***
101	118,6	536,0	77,86%	**238**	165,7	515,8	67,88%
103	120,0	529,9	77,35%	**239**	25,3	37,3	32,26%
104	117,5	541,9	78,31%	**240**	76,8	171,7	55,27%
201	113,8	582,7	80,47%	**241**	22,1	38,1	41,82%
202	72,1	276,9	73,96%	**246**	27,4	40,2	31,89%
203	70,9	294,9	75,97%	**247**	75,0	159,8	53,08%
204	118,9	523,2	77,28%	**248**	69,6	287,4	75,78%
205	115,5	2037,8	94,33%	**249**	73,7	308,0	76,06%
206	115,6	1255,3	90,79%	**250**	18,7	37,1	49,60%
207	118,2	546,2	78,37%	**251**	69,8	269,4	74,10%
208	73,9	274,0	73,04%	**252**	100,5	297,9	66,26%
209	71,9	256,6	72,00%	**253**	70,0	278,6	74,87%
210	72,2	637,5	88,67%	**254**	12,8	31,6	59,67%
221	104,0	414,9	74,93%	**257**	19,2	33,1	41,99%
222	112,6	403,2	72,07%	**258**	70,5	269,1	73,80%
223	163,3	483,5	66,23%	**259**	107,5	304,7	64,72%
224	120,3	547,4	78,02%	**260**	15,1	28,7	47,24%
225	118,5	554,8	78,63%	**261**	45,6	119,7	61,92%
228	32,1	59,4	46,00%	**262**	12,0	32,1	62,51%
230	95,3	354,9	73,14%	**265**	14,9	27,2	45,29%
231	118,1	532,1	77,81%	**266**	45,5	120,0	62,06%
232	106,4	463,6	77,05%	**301**	78,4	137,1	42,80%
233	21,1	40,0	47,31%	**302**	42,3	56,0	24,46%
236	32,4	53,5	39,51%	**303**	155,1	485,0	68,01%
237	113,3	439,1	74,20%	**304**	102,7	410,6	74,99%

6.2 Comparison between Memetic Ontology Alignment Systems and the State of the Art

This section is devoted to describe the statistical comparison carried out between our proposals and the existing ontology alignment systems. The whole experi-

mental session involves the standard benchmark track[1] provided by OAEI (see section 2.3.2.1). In our experimental scenario, F-measure (see definition 7) is used for evaluating the quality of produced alignments. The compared ontology alignment systems are the participants in OAEI competition 2010, whose F-measure values of the alignments produced for the standard benchmark track are well-known[2]. In particular, they are AgrMaker [26], AROMA [28], ASMOV [76], CODI [101], Eff2Match [137], Falcon [69], GeRMeSMB [110], MapPSO [16], RiMOM [136], SOBOM [142], TaxoMap [65]. Moreover, in the comparison, also a baseline ontology alignment system, named edna, using only an edit distance algorithm on labels is considered [73]. According to [40], ASMOV represents the current top-performer as for the standard benchmark track. The F-measure values for our proposals are computed by running the algorithms with configurations reported in Tables 6.1 and 6.2.

In detail, the comparison has involved two steps: in the first one, a statistical technique such as the Friedman's test [47] is used to determine whether the performances provided by the considered ontology alignment systems present any inequality; in the second one, which is performed only if in the first step an inequality is found, a post-hoc test such as Holm's test [68] is led in order to determined which ontology alignment system better outperforms.

Hereafter, the precise details about this statistical multiple comparison procedure are provided.

6.2.1 Friedman's test results

Friedman's test is a non-parametric statistical procedure which aims at detecting if a significant difference among the behavior of two or more algorithms exists. In particular, under the null-hypothesis, it states that all algorithms are equivalent, hence, a rejection of this hypothesis implies the existence of differences among the performance of all studied algorithms [49]. In order to reject the null hypothesis, the computed value χ_r^2 must be equal to or greater than the tabled critical chi-square value at the specified level of significance [116]. See section 3.7.2 for a

[1] http://oaei.ontologymatching.org/2010/
[2] http://oaei.ontologymatching.org/2010/results/

deeper description.

In our experimentation, a level of significance α equal to 0.05 is chosen. Since in our case we are comparing 14 algorithms $(12+2)$, our analysis has to consider the critical value $\chi^2_{0,05}$ for thirteen degrees of freedom that is equal to $22,36$. The sample data (see Table 6.5) for each ontology alignment system is represented by the F-measure values computed for each benchmark test case. In particular, the reported values represent the average on ten runs. In case of not computable F-measure (when both precision and recall are zero), a value equal to zero has been considered.

Table 6.6 shows the ranking obtained for each compared ontology alignment system. The computed Friedman's statistics is $\chi^2_r = 247,76$. Since it is greater than its associated critical value $\chi^2_{0,05} = 22,36$, the null hypothesis is rejected and it is possible to assess that there is a significant difference between at least two of the compared ontology alignment systems.

Attending to this result, a post-hoc statistical analysis is needed to conduct pairwise comparisons in order to detect concrete differences among compared ontology alignment systems and presented below.

6.2.2 Holm's test results

Holm's procedure is a multiple comparison procedure that works by setting a control algorithm and comparing it with the remaining ones. Normally, the algorithm which obtains the lowest value of ranking in the Friedman's test is chosen as control algorithm. In our experimentation, as shown in the Table 6.6, the algorithm with the lowest value of ranking is ASMOV.

Holm's test works on a family of hypotheses where each one is related to a comparison between the control method and one of the remaining algorithms. In details, the test statistic for comparing the i^{th} and j^{th} algorithm named z value is used for finding the corresponding probability from the table of the normal distribution (the so-called p-value), which is then compared with an appropriate level of significance α [31], in our experimentation equal to 0.05.

All data computed by the performed Holm's procedure are depicted in Table 6.7. By analysing the data, Holm's procedure rejects hypothesis from 1 to 11.

Table 6.5: Samples for Friedman's test. Each value represents the F-measure value computed for each benchmark test case. For sake of presentation, MOM stands for *MemeOptiMap* and MMM stands for *MemeMetaMap*.

No.	edna	AgrMaker	AROMA	ASMOV	CODI	Ef2Match	Falcon	GeRMeSMB	MapPSO	RiMOM	SOBOM	TaxoMap	MOM	MMM
101	1,00	0,99	0,98	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,51	0,99	1
103	1,00	0,99	0,99	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,51	1	1
104	1,00	0,99	0,99	1,00	0,99	1,00	1,00	1,00	1,00	1,00	1,00	0,51	1	1
201	0,04	0,92	0,95	1,00	0,13	0,77	0,97	0,94	0,42	1,00	0,95	0,51	0,97	0,95
202	0,03	0,89	0,00	0,88	0,00	0,08	0,00	0,39	0,05	0,81	0,64	0,02	0,35	0,44
203	1,00	0,98	0,80	1,00	0,86	1,00	1,00	0,98	1,00	1,00	1,00	0,49	1	1
204	0,93	0,97	0,97	1,00	0,74	0,99	0,96	0,98	0,98	1,00	0,99	0,51	0,99	1
205	0,34	0,92	0,95	0,99	0,28	0,84	0,97	0,99	0,73	0,99	0,96	0,51	0,91	0,99
206	0,54	0,93	0,95	0,99	0,39	0,87	0,94	0,92	0,85	0,99	0,96	0,51	0,96	0,96
207	0,54	0,93	0,95	0,99	0,42	0,87	0,96	0,96	0,81	0,99	0,96	0,51	0,97	0,96
208	0,93	0,96	0,58	1,00	0,61	0,95	0,98	0,95	0,79	1,00	1,00	0,44	0,98	0,98
209	0,35	0,88	0,37	0,92	0,22	0,47	0,65	0,59	0,16	0,87	0,71	0,14	0,64	0,68
210	0,54	0,93	0,18	0,96	0,24	0,38	0,66	0,58	0,32	0,85	0,82	0,15	0,72	0,76
221	1,00	0,97	0,99	1,00	0,98	1,00	1,00	1,00	1,00	1,00	1,00	0,51	1	1
222	0,98	0,98	0,99	1,00	1,00	1,00	1,00	0,99	1,00	1,00	1,00	0,46	1	1
223	1,00	0,95	0,93	1,00	1,00	1,00	1,00	0,96	0,98	0,98	0,99	0,45	1	1
224	1,00	0,99	0,97	1,00	1,00	1,00	0,99	1,00	1,00	1,00	1,00	0,51	1	1
225	1,00	0,99	0,99	1,00	0,99	1,00	1,00	1,00	1,00	1,00	1,00	0,51	1	1
228	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1	1
230	0,85	0,90	0,93	0,97	0,98	0,97	0,97	0,94	0,98	0,97	0,97	0,49	0,96	0,97
231	1,00	0,99	0,98	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,51	0,99	1
232	1,00	0,97	0,97	1,00	0,97	1,00	0,99	1,00	1,00	1,00	1,00	0,51	1	1
233	1,00	1,00	1,00	1,00	0,94	1,00	1,00	0,98	1,00	1,00	1,00	1,00	1	1
236	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1	1
237	0,98	0,98	0,97	1,00	0,99	1,00	0,99	1,00	0,99	1,00	1,00	0,46	1	1
238	1,00	0,94	0,92	1,00	0,99	1,00	0,99	0,96	0,97	0,98	0,98	0,45	1	1
239	0,50	0,98	0,98	0,98	0,98	0,98	1,00	0,98	0,98	0,98	0,98	0,94	0,98	0,98
240	0,55	0,91	0,83	0,98	0,95	0,98	1,00	0,85	0,92	0,94	0,98	0,88	0,77	0,96
241	1,00	1,00	0,98	1,00	0,94	1,00	1,00	0,98	1,00	1,00	1,00	1,00	1	1
246	0,50	0,98	0,97	0,98	0,98	0,98	1,00	0,98	0,98	0,98	0,95	0,94	0,98	0,98
247	0,55	0,88	0,80	0,98	0,98	0,98	1,00	0,91	0,89	0,94	0,98	0,88	0,76	0,96
248	0,03	0,72	0,00	0,87	0,00	0,02	0,00	0,37	0,05	0,64	0,48	0,02	0,25	0,35
249	0,03	0,88	0,02	0,88	0,02	0,08	0,00	0,35	0,05	0,78	0,64	0,02	0,35	0,44
250	0,02	0,56	0,00	0,62	0,00	0,11	0,00	0,06	0,03	0,73	0,20	0,00	0,42	0,48
251	0,05	0,78	0,00	0,86	0,00	0,08	0,00	0,44	0,06	0,68	0,47	0,02	0,38	0,36
252	0,02	0,78	0,00	0,86	0,00	0,08	0,00	0,37	0,02	0,68	0,50	0,02	0,3	0,32
253	0,03	0,72	0,02	0,87	0,02	0,02	0,00	0,42	0,07	0,61	0,47	0,02	0,27	0,34
254	0,02	0,56	0,00	0,43	0,00	0,00	0,00	0,06	0,03	0,56	0,00	0,00	0,16	0,3
257	0,00	0,56	0,00	0,50	0,00	0,11	0,00	0,06	0,07	0,71	0,25	0,00	0,42	0,43
258	0,04	0,78	0,02	0,86	0,02	0,08	0,00	0,28	0,07	0,66	0,47	0,02	0,38	0,36
259	0,02	0,78	0,02	0,86	0,02	0,08	0,00	0,29	0,04	0,64	0,50	0,02	0,3	0,28
260	0,00	0,61	0,00	0,57	0,00	0,13	0,00	0,00	0,12	0,67	0,06	0,00	0,49	0,5
261	0,00	0,54	0,00	0,57	0,00	0,11	0,00	0,06	0,03	0,55	0,09	0,00	0,31	0,33
262	0,05	0,56	0,00	0,43	0,00	0,00	0,00	0,06	0,04	0,53	0,00	0,00	0,16	0,27
265	0,03	0,61	0,00	0,46	0,00	0,13	0,00	0,00	0,04	0,63	0,06	0,00	0,49	0,47
266	0,00	0,54	0,00	0,44	0,00	0,11	0,00	0,06	0,06	0,52	0,05	0,00	0,31	0,3
301	0,59	0,59	0,73	0,86	0,38	0,71	0,78	0,71	0,64	0,73	0,84	0,43	0,83	0,87
302	0,43	0,32	0,35	0,73	0,59	0,71	0,71	0,41	0,04	0,73	0,74	0,40	0,64	0,64
303	0,00	0,78	0,59	0,83	0,65	0,83	0,77	0,00	0,00	0,86	0,50	0,36	0,68	0,79
304	0,83	0,86	0,84	0,95	0,74	0,95	0,94	0,77	0,72	0,94	0,91	0,52	0,89	0,93

Table 6.6: Rankings obtained through Friedman's test for each one of the compared ontology alignment systems

System	***Rank***
edna	9,48
AgrMaker	6,97
AROMA	11,02
ASMOV	**3,43**
CODI	10,28
Ef2Match	6,76
Falcon	7,87
GeRMeSMB	7,73
MapPSO	8,27
RiMOM	4,04
SOBOM	5,75
TaxoMap	12,17
MemeMetaMap	5,09
MemeOptiMap	6,14

Table 6.7: Holm's test

i	***System***	z ***value***	***unadjusted p-value***	$\alpha/(k-i)$, $\alpha = 0,05$
1	TaxoMap	10,4463	$1,5236 \cdot 10^{-25}$	0,0038
2	AROMA	9,0718	$1,1708 \cdot 10^{-19}$	0,0042
3	CODI	8,1873	$2,6712 \cdot 10^{-16}$	0,0045
4	edna	7,2311	$4,7898 \cdot 10^{-13}$	0,0050
5	MapPSO	5,7849	$7,2553 \cdot 10^{-9}$	0,0056
6	Falcon	5,3068	$1,1156 \cdot 10^{-7}$	0,0063
7	GeRMeSMB	5,1395	$2,7550 \cdot 10^{-7}$	0,0071
8	AgrMaker	4,2311	$2,3254 \cdot 10^{-5}$	0,0083
9	Ef2Match	3,98019	$6,8883 \cdot 10^{-5}$	0,0100
10	MemeOptiMap	3,2391	0,0012	0,0125
11	SOBOM	2.,7729	0,0056	0,0167
12	MemeMetaMap	1,9841	0,0472	0,0250
13	RiMOM	0,7291	0,4659	0,0500

As a consequence, it is possible to state that ASMOV statistically outperforms TaxoMap, AROMA, CODI, edna, MapPSO, Falcon, GeRMeSMB, AgrMaker, Ef2Match, *MemeOptiMap* and SOBOM at 5% significance level but not RiMOM and *MemeMetaMap*.

By concluding, the statistical comparison states that *MemeMetaMap* has the same performances of ASMOV, which is, as declared by OAEI [40], currently the top-performer as for the standard benchmark track. In addition, as described in Sect. 5.2.1.2, our system has the significant advantage of not requiring *a priori* information about ontologies under alignment and data availability, unlike other existing approaches including ASMOV (as described in Sect. 2.4.2).

Chapter 7

Conclusions and Future Works

In this chapter, we conclude this thesis by, firstly, giving a summary with the contributions of our research work (see section 7.1), and, then, discussing some ideas for future works (see section 7.2).

7.1 Summary

In this thesis, we present our research work about the *ontology alignment problem*, i.e., the issue to identify semantic matchings between heterogeneous ontologies. Finding solutions which solve this problem is very important task in several industrial and academic application domains such as knowledge management, information retrieval, medical diagnosis, e-Commerce, knowledge acquisition, search engines, bioinformatics, the emerging Semantic Web and so on. However, in spite of the enormous number of ontology alignment systems which have been developed in order to face this problem, there is no an integrated solution that is a clear success, which is robust enough to be the basis for future development, and which is usable by non expert users [117]. Starting from this consideration, we have researched a new innovative approach to address this relevant problem consisting in the exploitation of an emergent class of evolutionary algorithms, named *memetic algorithms*.

In particular, in this thesis, firstly, we have studied the ontology alignment problem and all its aspects: its relevance, its causes, the current techniques used to address it, the open issues (Chapter 2). Successively, understood the complex

nature of the problem, we are interested for its resolution in a class of approximation methods, i.e., the memetic algorithms, with aim of producing sub-optimal alignments. After a detailed study of memetic algorithms (Chapter 3), we have developed two ontology alignment systems based on this category of algorithms, named *MemeOptiMap* and *MemeMetaMap*, which address the ontology alignment problem by two different point of views. In detail, the former, presented in Chapter 4, uses memetic algorithms to directly solve the ontology alignment problem as a minimum optimization problem, whereas, the latter, presented in Chapter 5, exploits memetic algorithms to address meta-matching problem, i.e., the issue related to determining the appropriate values for ontology alignment process parameters and, consequently, it produces an alignment by performing a typical matching characterized by the computed parameters.

Both systems are the result of an evolving development process having the aim of making changes in order to achieve relevant improvements in terms of both alignment quality and effort performance. In detail, the process, initially, has involved, for both systems, the definition of all their basic modules which are mainly related to the elementary components of a memetic algorithm: chromosome structure, fitness function and integrated local search procedures. Then, we have studied the initial versions of developed systems in order to identify their weaknesses. During this phase, in particular, we found that *MemeOptiMap* was strongly affected by the high computational cost of directly optimizing the ontology alignment problem, whereas, *MemeMetaMap*'s behavior was highly dependent by specific instance parameters. Hence, the current final version of *MemeOptiMap* is based on a hybrid parallel model, named *multi-island parallel memetic algorithm*, which allows to distribute ontology alignment task across multiple processing nodes, and, as a consequence, speed up the search process and to preserve population diversity useful to increase the chances of escaping from local optima. As for *MemeMetaMap*, in its current final version, it has been enhanced with an expert fuzzy systems capable of adaptively regulating the specific instance parameters affecting its behaviour by means of the analysis of the features of ontologies composing the alignment task at hand.

In Chapter 6, both systems in their final versions have been compared with the state of the art by means of a statistical multiple comparison procedure.

The test results show that both approaches are competitive, and, in particular, *MemeMetaMap* improves the capabilities of the current ontology alignment processes by working regardless of the user involvement, data availability and the need of a priori knowledge about ontology features, and, yielding high performance in terms of alignment quality with respect to top-performers of well-known OAEI[1].

7.2 Future works

Our future work lies in extension of the implemented ontology alignment systems, *MemeOptiMap* and *MemeMetaMap*, in order to furthermore improve their performance both in terms of alignment quality and computational effort. Regarding *MemeOptiMap*, we plan to improve both computational cost and alignment quality by actually implementing a multi-agent system composed of more numerous optimizing agents in order to massively distribute computational effort and strongly increase benefits provided by the migration process. Moreover, by analiysing this approach, it suffers from the so-called meta-matching problem, i.e., determining ontology alignment parameters that should be properly set to get the best possible match results. This drawback can be faced by supporting *MemeOptiMap* with machine learning techniques or evolutionary approaches such as that investigated in our second approach. As for *MemeMetaMap*, due to the composite nature of the exploited fitness function, we plan to investigate the possibility to improve alignment quality by performing a multi-objective optimization approach in order to simultaneously optimize the two considered objectives: the number of correspondences to be found and the similarity average. Besides, in order to improve computational cost, an idea could be to distribute *MemeMetaMap* by using a global parallel strategy capable of organizing the computation of fitness functions, which is the most effort, over several nodes. Finally, in order to further improve the quality of produced alignments for both systems, our idea is to enable the proposed ontology alignment systems to use similarity measures characterized by a higher semantic power and to detect not only equivalence mappings but

[1] http://oaei.ontologymatching.org/

also other kinds of correspondences based on relations such as subsumption and mismatch.

References

[1] S. Abdullah and H. Turabieh. On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. *Inf. Sci.*, **191**:146–168, May 2012. 70

[2] G. Acampora, P. Avella, V. Loia, S. Salerno, and A. Vitiello. Improving ontology alignment through memetic algorithms. In *FUZZ-IEEE 2011, IEEE International Conference on Fuzzy Systems*, pages 1783–1790, 2011. 77

[3] G. Acampora, M. Gaeta, E. Muñoz Ballester, and A. Vitiello. An adaptive multi-agent memetic system for personalizing e-learning experiences. pages 123–130, 2011. 70

[4] G. Acampora, U. Kaymak, V. Loia, and A. Vitiello. Hybridizing genetic algorithms and hill climbing for similarity aggregation in ontology matching. In *Computational Intelligence (UKCI), 2012 12th UK Workshop on*, pages 1 –6, sept. 2012. 106

[5] G. Acampora and V. Loia. Fuzzy control interoperability and scalability for adaptive domotic framework. *Industrial Informatics, IEEE Transactions on*, **1**[2]:97 – 111, may 2005. 123

[6] G. Acampora, V. Loia, C.-S. Lee, and M.-H. Wang, editors. *On the Power of Fuzzy Markup Language*. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2013. 123

[7] G. Acampora, V. Loia, S. Salerno, and A. Vitiello. A hybrid evolutionary approach for solving the ontology alignment problem. *Int. J. Intell. Syst.*, **27**[3]:189–216, 2012. 77

[8] G. Acampora, V. Loia, and A. Vitiello. Hybridizing fuzzy control and timed automata for modeling variable structure fuzzy systems. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1 –8, july 2010. 121

[9] G. Acampora, V. Loia, and A. Vitiello. An enhanced visual environment for designing, testing and developing fml-based fuzzy systems. In G. Acampora, V. Loia, C.-S. Lee, and M.-H. Wang, editors, *On the Power of Fuzzy Markup Language*, Studies in Fuzziness and Soft Computing, pages 17–31. Springer Berlin Heidelberg, 2013. 123

[10] G. Acampora and A. Vitiello. Improving agent interoperability through a memetic ontology alignment: A comparative study. In *FUZZ-IEEE 2012, IEEE International Conference on Fuzzy Systems*, pages 1–8, 2012. 89

[11] G. Acampora and A. Vitiello. Collaborative memetic agents for enabling semantic interoperability. In *2013 IEEE Symposium Series on Computational Intelligence (SSCI 2013)*, April 2013. 95

[12] G. Antoniou and F. van Harmelen. Web ontology language: Owl. In S. Staab and R. Studer, editors, *The Handbook on Ontologies*. Springer, 2004. 14

[13] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996. iv, 4

[14] V. Bertaud-Gounot, R. Duvauferrier, and A. Burgun. Ontology and medical diagnosis. *Inform Health Soc Care*, **37**[1]:22–32, 2012. 18

[15] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, **35**[3]:268–308, September 2003. 62

[16] Jurgen Bock and Jan Hettenhausen. Discrete particle swarm optimisation for ontology alignment. *Information Sciences*, **192**[0]:152 – 173, 2012. 51, 106, 110, 135

[17] W. Borst. *Construction of Engineering Ontologies*. PhD thesis, University of Tweenty, Enschede, The Netherlands, 1997. 8

[18] G. Budai-Balke, R. Dekker, and U. Kaymak. Genetic and memetic algorithms for scheduling railway maintenance activities. Econometric Institute Report EI 2009-30, Erasmus University Rotterdam, Econometric Institute, 2009. 70

[19] E. K. Burke and A. J. Smith. A Memetic Algorithm to Schedule Planned Grid Maintenance. *ACM Journal of Experimental Algorithms*, 1999. 70

[20] E. Cantú-Paz. A survey of parallel genetic algorithms. *CALCULATEURS PARALLELES*, **10**, 1998. 71

[21] E. Cantú-Paz. On the effects of migration on the fitness distribution of parallel evolutionary algorithms. In *Workshop on Evolutionary Computation and Parallel Processing at GECCO 2000*, pages 3–6, 2000. 100

[22] R.G. Castro, D. Maynard, D. Foxvog, H. Wache, and R. Gonzlez-Cabero. D2.1.4: Specification of a methodology, general criteria, and benchmark suites for benchmarking ontology tools. Technical report, Knowledge web NoE, 2004. 35

[23] C. H. Che, Z. Zhang, and A. Lim. A memetic algorithm for solving multiperiod vehicle routing problem with profit. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 45–46, New York, NY, USA, 2011. ACM. 70

[24] C. Cotta, E. Alba, and J. M. Troya. Stochastic reverse hillclimbing and iterated local search. In *In Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1558–1565. IEEE, 1999. 60

[25] C. Cotta, A. Mendes, V. Garcia, P. França, and P. Moscato. Applying memetic algorithms to the analysis of microarray data. In *Proceedings of the 2003 international conference on Applications of evolutionary computing*, EvoWorkshops'03, pages 22–32, Berlin, Heidelberg, 2003. Springer-Verlag. 70

[26] I. F. Cruz, C. Stroe, M. Caci, F. Caimi, M. Palmonari, F. Palandri Antonelli, and U. C. Keles. Using agreementmaker to align ontologies for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 135

[27] R. Qing dao-er ji and Y. Wang. A new hybrid genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, **39**[10]:2291 – 2299, 2012. 69

[28] J. David. *AROMA: une méthode pour la découverte d'alignements orientés entre ontologies à partir de règles d'association.* PhD thesis, Université de Nantes, France, 2007. 135

[29] R. Dawkins. *The Selfish Gene.* Oxford University Press, September 1990. 56, 67, 78

[30] K.A. De Jong. Evolving intelligent agents: A 50 year quest. *Computational Intelligence Magazine, IEEE*, **3**[1]:12 –17, february 2008. 95

[31] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, **7**:1–30, December 2006. 74, 75, 93, 136

[32] J. Digalakis, K. Margaritis, and K. G. Margaritis. A performance comparison of parallel genetic and memetic algorithms using mpi. Technical report, Distributed Processing Laboratory, University of Macedonia, 2000. 95

[33] H.-H. Do and E. Rahm. Matching large schemas: Approaches and evaluation. *Inf. Syst.*, **32**[6]:857–885, September 2007. 49

[34] A.i Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies in Information Systems*, pages 397–416, 2003. 50

[35] C. Eccher, A. Ferro, and D.M. Pisanelli. An ontology of therapies. In Patty Kostkova, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Electronic Healthcare*, **27** of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 139–146. Springer Berlin Heidelberg, 2010. 18

[36] Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap*, **4** of *Semantic Web And Beyond Computing for Human Experience*. Springer, 2007. 10, 11, 14, 25, 43

[37] J. Euzenat. An api for ontology alignment. In *Proc. 3rd international semantic web conference*, pages 698–712, 2004. 40

[38] J. Euzenat, T. Le Bach, J. Barrasa, P. Bouquet, J. De Bo, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, and et al. State of the art on ontology alignment. In *Deliverable D2.2.3 v1.2. Knowledge Web*, 2008. 25, 40

[39] J. Euzenat, R. Garca Castro, and M. Ehrig. D2.2.2 specification of a benchmarking methodology for alignment techniques. Technical report, KnowledgeWeb Networkl of Excellence, 2005. 38

[40] J. Euzenat, A. Ferrara, W. Robert van Hage, L. Hollink, C. Meilicke, A. Nikolov, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, and C. Trojahn. Final results of the ontology alignment evaluation initiative 2011. In *Proceedings of the 6th International Workshop on Ontology Matching*, 2011. 52, 135, 139

[41] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, and C. Trojahn dos Santos. Ontology alignment evaluation initiative: Six years of experience. *J. Data Semantics*, **15**:158–192, 2011. 35, 36

[42] J. Euzenat, F. Scharffe, and L. Serafini. Specification of the delivery alignment format. In *Deliverable D2.2.6. Knowledge Web*, 2006. 39

[43] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Heidelberg, 2007. 10, 11, 19, 23, 83, 84, 114, 115

[44] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001. 13, 18

[45] Muller J.P.-Muller J. Odell J. Berre A.J. Fischer, K. Agent-based technologies and applications for enterprise interoperability. *Lecture Notes in Business Information Processing*, **5**, 2009. 3, 94

[46] C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, **63**:437–461, 1996. 69

[47] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, **32**[200]:pp. 675–701, 1937. 72, 135

[48] L. Gao, G. Zhang, L. Zhang, and X. Li. An efficient memetic algorithm for solving the job shop scheduling problem. *Comput. Ind. Eng.*, **60**[4]:699–705, May 2011. 69

[49] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, **15**[6]:617–644, December 2009. 72, 74, 91, 135

[50] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. iii, 2

[51] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA, 1987. 8

[52] J.H. Gennari, S.W. Tu, T.E. Rothenfluh, and M.A. Musen. Mappings domains to methods in support of reuse. *Int. J. on Human-Computer Studies*, **41**:399–424, 1994. 9

[53] A.-L. Ginsca and A. Iftene. Using a genetic algorithm for optimizing the similarity aggregation step in the process of ontology alignment. In *Roedunet International Conference (RoEduNet)*, pages 118–122, 2010. 51, 54, 110

[54] F. Giunchiglia, M. Yatskevich, P. Avesani, and P. Shivaiko. A large dataset for the evaluation of ontology matching. *Knowl. Eng. Rev.*, **24**[2]:137–157, June 2009. 36

[55] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002. 83

[56] A. Gómez-Pérez and O. Corcho. Ontology languages for the semantic web. *IEEE Intelligent Systems*, **17**[1]:54–60, 2002. 14

[57] J. Gracia, J. Bernard, and E. Mena. Ontology matching with cider: evaluation report for oaei 2011. In *Proceedings of the 6th International Workshop on Ontology Matching*, 2011. 49

[58] B. C. Grau, I. Horrocks, B. Parsia, P. Patel-Schneider, and U. Sattler. Next steps for owl. *OWL Experienced and Directions*, 2006. 14

[59] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies, version 3.0. *Technical Report, KSL 91-66, Knowledge Systems Laboratory, Department of Computer Science*, 1992. 13

[60] T. R. Gruber. *Towards principles for the design of ontologies used for knowledge sharing*. Formal Ontology in Conceptual Analysis and Knowledge

Representation. Kluwer Academic Publishers, Deventer, The Netherlands, 1993. iii, 2, 8

[61] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, **5**[2]:199–220, 1993. 8, 9, 10

[62] N. Guarino, D. Oberle, and S. Staab. What is an ontology? In *The Handbook on Ontologies*. Springer-Verlag, 2009. 7, 8

[63] G. Gutin and D. Karapetyan. A memetic algorithm for the generalized traveling salesman problem. **9**[1]:47–60, March 2010. 69

[64] A. B. Hadj-Alouane, J. C. Bean, and K. G. Murty. A hybrid genetic/optimization algorithm for a task allocation problem. *Journal of Scheduling*, **2**:189–201, 1999. 70

[65] F. Hamdi, B. Safar, N. B. Niraula, and C. Reynaud. Taxomap alignment and refinement modules: Results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 135

[66] P. J. Hayes. Naive physics: Ontology for liquids. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 71–107. Norwood, New Jersey: Ablex Publishing Corporation, 1985. 18

[67] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975. iv, 4, 64, 65

[68] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, **6**[2]:pp. 65–70, 1979. 72, 135

[69] W. Hu, J. Chen, G. Cheng, and Y. Qu. Objectcoref & falcon-ao: Results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 135

[70] J. Huang, J. Dang, J. M. Vidal, and M. N. Huhns. Ontology matching using an artificial neural network to learn weights. In *IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition*, 2007. 106

[71] T. C. Hughes and B. C. Ashpole. The Semantics of Ontology Alignment. In *I3CON. Information Interpretation and Integration Conference*, 2004. iv, 4

[72] Tang J., M. H. Lim, and Y. S. Ong. A parallel hybrid GA for combinatorial optimization using grid technology. In *IEEE Congress on Evolutionary Computation*, 2003. 71, 95

[73] C. Meilicke J. Pane F. Scharffe P. Shvaiko H. Stuckenschmidt O. Švab-Zamazal V. Svatek J. Euzenat, A. Ferrara and C. Trojahn dos Santos. Final results of the ontology alignment evaluation initiative 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 117, 135

[74] J. Jantzen. Design of fuzzy controllers. Technical report, Department of Automation, Technical University of Denmark, Denmark, 1998. 123

[75] M. A. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, **14**:491–498, 1995. 29

[76] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Ontology matching with semantic verification. *Web Semant.*, **7**[3]:235–251, September 2009. 49, 135

[77] T. Jones. Evolutionary algorithms, fitness landscapes and search. Working Papers 95-05-048, Santa Fe Institute, May 1995. 58

[78] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, **18**[1]:1–31, January 2003. iv, 3, 44

[79] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, **34**:975–986, 1984. 63

[80] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, **220, 4598**:671–680, 1983. 64

[81] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Trans. Evolutionary Computation*, **9**[5]:474–488, 2005. 68, 89

[82] D. Kumar, S. Kumar, and C. S. Rai. Memetic algorithms for feature selection in face recognition. In *Proceedings of the 2008 8th International Conference on Hybrid Intelligent Systems*, HIS '08, pages 931–934, Washington, DC, USA, 2008. IEEE Computer Society. 70

[83] C.C. Lee. Fuzzy logic in control system: Fuzzy logic controller - part ii. *IEEE Transactions Systems, Man & Cybernetics*, **20**[2]:404–435, march 1990. 123

[84] I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, **10**:707, 1966. 28

[85] B. Liu, L. Wang, Y. Jin, and D. Huang. Designing neural networks using pso-based memetic algorithm. In *Proceedings of the 4th international symposium on Neural Networks: Advances in Neural Networks, Part III*, ISNN '07, pages 219–224, Berlin, Heidelberg, 2007. Springer-Verlag. 18, 70

[86] Z. Lü and J.-K. Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, **203**[1]:241 – 250, 2010. 69

[87] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 49

[88] E. H. Mamdani. Applications of fuzzy algorithms for simple dynamic plants. *Proceedings of IEE*, **121**[12]:15851588, 1974. 119

[89] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, **7**[1]:1 – 13, 1975. 123

[90] J. Martinez-Gil, E. Alba, and J. F. A. Montes. Optimizing Ontology Alignments by Using Genetic Algorithms. In Christophe Guéret, Pascal Hitzler, and Stefan Schlobach, editors, *Nature inspired Reasoning for the Semantic Web (NatuReS)*, **419**. CEUR Workshop Proceedings, October 2008. iv, 4, 51, 106

[91] J. Martinez-Gil and J. F. Aldana-Montes. Evaluation of two heuristic approaches to solve the ontology meta-matching problem. *Knowl. Inf. Syst.*, **26**[2]:225–247, February 2011. 26, 106

[92] P. Merz and B. Freisleben. Memetic algorithms for the traveling salesman problem. *Complex Systems*, **13**:297–345, 1997. 69

[93] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 77–88, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 53

[94] D. Milward, M. Bjäreland, W. Hayes, M. Maxwell, L. Öberg, N. Tilford, J. Thomas, R. Hale, S. Knight, and J. Barnes. Ontology-based interactive information extraction from scientific abstracts: Conference papers. *Comp. Funct. Genomics*, **6**[1-2]:67–71, February 2005. 18

[95] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms, 1989. 56

[96] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, 2003. 55, 56, 57, 58

[97] Y. Nagata and O. Bräysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Netw.*, **54**[4]:205–215, December 2009. 70

[98] J.M. Vázquez Naya, M. Martínez Romero, J.P. Loureiro, C.R. Munteanu, and A. Pazos Sierra. Improving ontology alignment

through genetic algorithms. In *Soft Computing Methods for Practical Environment Solutions: Techniques and Studies*, pages 240–259. IGI Global, Hershey, PA, USA, 2010. 51, 54, 110

[99] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, **2**[0]:1 – 14, 2012. 67

[100] F. Neri, N. Kotilainen, and M. Vapa. An adaptive global-local memetic algorithm to discover resources in p2p networks. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog: Applications of Evolutionary Computing*, pages 61–70, Berlin, Heidelberg, 2007. Springer-Verlag. 70

[101] J. Noessner and M. Niepert. Codi: Combinatorial optimization for data integration results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 49, 135

[102] M. G. Norman and P. Moscato. A competitive and cooperative approach to complex combinatorial search. Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA, 1989. 67

[103] N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press, 2000. 49

[104] Y. S. Ong, Q. H Nguyen, M. H. Lim, and T. Jing. A development platform for memetic algorithm design. In *Joint 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on Advanced Intelligent Systems*, 2006. 66

[105] E. Franconi L. Serafini G. Stamou S. Tessaris P. Bouquet, J. Euzenat. D2.2.1: Specification of a common framework for characterizing alignment. Technical report, NoE Knowledge Web project delivable, 2004. 26

[106] M. Paolucci, T. Kawamura, T.R. Payne, and K.P. Sycara. Semantic matching of web services capabilities. In *Proceedings of International Semantic Web Conference*, pages 333–347, 2002. 18

[107] S. Pavel and J. Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, **99**[PrePrints], 2012. iv, 22, 105

[108] H. Sofia Pinto, A. Gomez-Perez, and J. P. Martins. Some issues on ontology integration. In *Proc. of IJCAI99s Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, 1999. 21, 22

[109] V. Qazvinian, H. Abolhassani, S. H. Haeri, and B. B. Hariri. Evolutionary coincidence-based ontology mapping extraction. *Expert Systems*, **25**[3]:221–236, 2008. 51

[110] C. Quix, A. Gal, T. Sagi, and D. Kensche. An integrated matching system: Geromesuite and smb results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 135

[111] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, **10**[4]:334–350, December 2001. 44

[112] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, **10**[4]:334–350, December 2001. 25

[113] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. 38

[114] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, **24**[5]:513–523, August 1988. 30

[115] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, **18**[11]:613–620, November 1975. 30

[116] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 4 edition, 2007. 72, 74, 75, 92, 135

[117] P. Shvaiko and J. Euzenat. Ten challenges for ontology matching. In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems*, OTM '08, pages 1164–1182, Berlin, Heidelberg, 2008. Springer-Verlag. 3, 52, 140

[118] P.l Shvaiko and J. Euzenat. A survey of schema-based matching approaches journal on data semantics iv. In Stefano Spaccapietra and Stefano Spaccapietra, editors, *Journal on Data Semantics IV*, **3730** of *Lecture Notes in Computer Science*, chapter 5, pages 146–171. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005. x, 44, 46, 47, 48, 49

[119] M. Srinivas and L. M. Patnaik. Genetic algorithms: A survey. *IEEE Comput.*, **27**[6]:17–26, 1994. 64, 65

[120] R. Stevens, C. Goble, and S. Bechhofer. Ontology-based Knowledge Representation for Bioinformatics. *Briefings in Bioinformatics.*, **1**[4]:398–414, 2000. 10

[121] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proceedings of the 4th international conference on The Semantic Web*, ISWC'05, pages 624–637, Berlin, Heidelberg, 2005. Springer-Verlag. 22, 29, 30, 105

[122] R. Studer, R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, **25**[1]. 8, 9

[123] O. Svab-Zamazal. *Pattern-based Ontology Matching and Ontology Alignment Evaluation*. PhD thesis, University of Economics, Prague, 2010. 12

[124] T. Taguchi, T. Yokota, and M. Gen. Reliability optimal design problem with interval coefficients using hybrid genetic algorithms. *Computers amp; Industrial Engineering*, **35**[1-2]:373 – 376, 1998. 69

[125] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions Systems, Man & Cybernetics*, **15**[1]:116–132, 1985. 121

[126] V. Tamma, S. Phelps, I. Dickinson, and M. Wooldridge. Ontologies for supporting negotiation in e-commerce. *Engineering Applications of Artificial Intelligence*, **18**[2]:223 – 236, 2005. 18

[127] J. Tang, M. H. Lim, and Y. S. Ong. Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. *Soft Comput.*, **11**[9]:873–888, April 2007. 69, 71

[128] J. Tang, M.H. Lim, Y.S. Ong, and M.J. Er. Study of migration topology in island model parallel hybrid-ga for large scale quadratic assignment problems. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, **3**, pages 2286–2291, dec. 2004. 100

[129] M. M. Taye. *Ontology Alignment Mechanisms for Improving Web-based Searching*. PhD thesis, De Montfort University, Leicester, UK, 2009. 12

[130] M. M. Taye. Understanding semantic web and ontologies: Theory and applications. *Journal of Computing*, **2**, 2010. 18

[131] C.-K. Ting and C.-C. Liao. A memetic algorithm for extending wireless sensor network lifetime. *Inf. Sci.*, **180**[24]:4818–4833, December 2010. 69

[132] P. E. van der Vet, P.-H. Speel, and N. J. I. Mars. Ontologies for very large knowledge bases in materials science: A case study. In N. J. I. Mars, editor, *TowardsVery Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 73–83. IOS Press, 1995. 18

[133] M. Vargas-Vera and M. Nagy. Towards intelligent ontology alignment systems for question answering: Challenges and roadblocks. *Journal of Emerging Technologies in Web Intelligence*, **2**[3]:244–257, 2010. 52

[134] H. Wache, T. V'ogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. H'ubner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 108–117, 2001. iv, 3, 44

[135] J. Wang, Z. Ding, and C. Jiang. Gaom: Genetic algorithm based ontology matching. In *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing*, APSCC '06, pages 617–620, Washington, DC, USA, 2006. IEEE Computer Society. iv, 4, 51, 106

[136] Z. Wang, X. Zhang, L. Hou, Y. Zhao, J. Li, Y. Qi, and J. Tang. Rimom results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 49, 105, 135

[137] W. Wei, K. Chua, and J.-J. Kim. Eff2match results for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 49, 135

[138] Y. Wen, H. Xu, and J. Yang. A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system. *Information Sciences*, **181**[3]:567 – 581, 2011. 70

[139] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, **1**[6]:pp. 80–83, 1945. 72

[140] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999. 29

[141] M. Wooldridge. *An Introduction to MultiAgent Systems - Second Edition*. John Wiley & Sons, Reading, MA, 2009. 95

[142] P. Xu, Y. Wang, L. Cheng, and T. Zang. Alignment results of sobom for oaei 2010. In *Proceedings of the 5th International Workshop on Ontology Matching*, 2010. 135

[143] R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on*, **18**[1]:183 –190, jan/feb 1988. 34

[144] J.H. Zar. *Biostatistical Analysis*. Prentice Hall, 1999. 74