Knowledge Compilation for Core Competence Extraction in Organizations

Simona Colucci[1], Eufemia Tinelli[2], Silvia Giannini[2], Eugenio Di Sciascio[2], and Francesco M. Donini[1]

[1] DISUCOM, Università della Tuscia, Viterbo, Italy
[2] SisInfLab & DEI, Politecnico di Bari, Bari, Italy

Abstract. Knowledge Management (KM) asks for information-intensive services over large amount of data, modeling the intellectual capital of an organization. To combine the expressiveness of logic-based languages with efficient information processing, we adopt a Knowledge Compilation approach to the extraction of "Core Competence" of a given company, a typical KM problem. In particular, we translate into a relational database schema the full logical description formalized in a Knowledge Base (KB), modeling organizational intellectual capital according to the formalism of Description Logics (DLs). Core Competence extraction is consequently performed through standard-SQL queries, while retaining the expressiveness of the logical representation. The service has been embedded in a system for Human Resource Management, `I.M.P.A.K.T.`[3], to show how Core Competence extraction performance significantly improves w.r.t. implementations exploiting DL reasoning engines.

Keywords: Description Logics, Core Competence Extraction, RDBMS, SQL

1 Introduction

Knowledge management in an organizational environment is a complex, heterogeneous and information-intensive task which deserves the design of specific and sophisticated services in order to be performed. KM-related processes may in fact range in a wide set of tasks, from strategic choices decision support to business activities allocation, just to name a few. The automation of such processes is usually achieved by integrated enterprise suites relying on database technologies[4], which are able to cope with scalability issues, but – if traditionally employed – do not allow for a machine-understandable representation of the full informative content to be managed. On the contrary, the peculiarity of knowledge as organizational asset asks for technologies facilitating both representation and processing of information, like semantic-based ones [1]. In this paper we show how to combine the advantages of both semantic-based and database technologies in a Knowledge Compilation [2] approach, devoted to the identification of Core Competence in organizational support contexts.

Hamel and Prahlad [3] define Core Competence as a sort of organizational capability providing customer benefits, hard to be imitated from competitors and possessing

[3] Information Management and Processing with the Aid of Knowledge-based Technologies
[4] `http://www.monster.com/`, `http://www.careerbuilder.com`

leverage potential. Among available approaches to strategic management (see [4] for a classification), our proposal takes the *competence-based perspective* ([5], [6]) and interprets company strategic competence as a collective asset, resulting from the synergy of human resources. We model the company intellectual capital in a Knowledge Base, which is described according to the formalism of DLs [7].

Coherently with the adopted Knowledge Compilation schema, our contribution makes computationally efficient Core Competence extraction over the information contained in the KB, by splitting the reasoning process in two phases: (i) the KB is pre-processed, thus parsing it in a specifically designed relational database schema (*off-line reasoning*); (ii) the querying process for Core Competence extraction is performed by exploiting the structure coming from the first phase (*on-line reasoning*). The approach has been implemented in `I.M.P.A.K.T.`, a system that manages skill matching [8] and team composition [9] tasks together with Core Competence extraction. `I.M.P.A.K.T.` implements via SQL queries standard plus non–monotonic and non–standard DLs inference services.

Other systems exploiting DBMS techniques to deal with reasoning tasks have been proposed in the literature (see *KAON2*[5], [10], [11], *QuOnto*[6] and *PelletDB*[7], among others). Even when their languages are more expressive than the one we use in our system, they are mostly able to return either exact matches (*i.e.*, instance retrieval) or general query answering. Instead, we use an enriched relational schema to deal with non-standard inferences to provide effective value-added services, including an approximate matching specifically fitting human resources management.

The paper is organized as follows: in the next Section we briefly recall both the DLs formalism and the adopted reasoning services, to make the paper self-contained. Section 3 details the Knowledge Compilation schema at the basis of the Core Competence extraction process, outlined in Section 4. The achieved performance improvements are presented in Section 5, through a comprehensive experimental evaluation. Finally, conclusions close the paper.

2 Background

Description Logics [7, Ch.2] are a family of formalisms and reasoning services for knowledge representation, whose alphabet is made up by unary and binary predicates, known as **Concept Names** and **Role Names**. Complex **Concept Descriptions** are built from concept and role names composed by *constructors*. Each choice of constructors defines a different DL, and characterizes it both in terms of expressiveness and computational complexity [7, Ch.3]. The expressiveness of a DL may be also enriched by the introduction of *concrete features*, which are binary predicates whose second argument belongs to a *concrete domain* D (*e.g.*, integers, reals, strings, dates). Given a DL $\mathcal{L}$, its enrichment with concrete features is denoted by $\mathcal{L}(\mathrm{D})$. Statements about classes in the domain of interest are divided into: **Concept Definitions** (denoted by $A \equiv C$) stating —in the form of a complex concept C—the necessary and sufficient conditions

[5] `http://kaon2.semanticweb.org/`

[6] `http://www.dis.uniroma1.it/~quonto/`

[7] `http://clarkparsia.com/pelletdb/`

for an individual to belong to the concept A; **Concept Inclusions** (denoted by $A \sqsubseteq C$) stating in C only the necessary conditions for membership in A. The set of inclusions and definitions formalize the intensional knowledge of the domain of interest, known as **TBox** in DL systems. A DL system usually allows one to make statements about named individuals, which make up the part of a DL-knowledge base known as **ABox**. **ABox** may include either **Concept assertions** ($C(a)$ states that an individual a belongs to the concept C) or **Role assertions** ($r(a, b)$ states that individual a relates to the individual b through role r).

In order to fully represent the features of Human Resources management, real-life examples suggest that at least the following constructors are needed: conjunction, universal and existential quantification, and concrete features (which define $\mathcal{ALE}(\mathrm{D})$). However, the interplay of existential and universal quantification leads to reasoning problems that are not computable in polynomial time [7, Ch.3], and such computational complexity hampers the translation into SQL of our problems.

Therefore, `I.M.P.A.K.T.` adopts a Curriculum Vitae (CV) representation (see Definition 2) allowing for reasoning only on $\mathcal{FL}_0(\mathrm{D})$ concepts. Such a DL drops existential quantification and thus gives up to the full $\mathcal{ALE}(\mathrm{D})$ expressiveness for reaching computability. The most important reasoning service in DL checks for specificity hierarchies, by determining whether a concept description is more specific than another one or, formally, if there is a *subsumption* relation between them. It is noteworthy that the framework underlying `I.M.P.A.K.T.` solves subsumption in $\mathcal{FL}_0(\mathrm{D})$ only via SQL queries, without reference to any exponential-time inference engine.

Although very useful in many knowledge management settings, subsumption does not allow to solve any emerging issue and non-standard reasoning services need to be specifically developed. In particular, the service category we here present aims at automatically extracting Core Competence, by identifying a common know-how in a significant portion of personnel, with such a portion cardinality to be set by the management. To this aim, our knowledge compilation approach framework follows the conceptual line in [12], based on Least Common Subsumer (LCS) computation.

By definition [13], for a collection of concept descriptions, their LCS represents the most specific concept description subsuming all of the elements of the collection. Nevertheless, in some applications, like Core Competence identification, such a sharing is not required to be full: the objective is finding a concept description subsuming *a portion of* the elements in a collection, rather than the collection as a whole. Such a concept description has been defined *k-Common Subsumer* (k-CS) [12]:

Definition 1. *Let $C_1, \ldots, C_n$ be n concepts in a DL $\mathcal{L}$, and let be $k < n$. A* ***k-Common Subsumer*** *(k-CS) of $C_1, \ldots, C_n$ is a concept D such that D is an LCS of k concepts among $C_1, \ldots, C_n$.*

Some k-Common Subsumers, defined *Informative k-Common Subsumers* ($IkCS$ [12]), are strictly subsumed by the collection LCS and then add informative content to it .

Among possible $IkCSs$, some subsume the biggest number of concepts in the collection and have therefore been defined as *Best Informative Common Subsumers* ($BICS$ [12]).

If a collection admits a meaningful LCS (*i.e.*, not equivalent to the universal concept $\top$), such LCS is the best common subsumer it may have. Else, for collections whose LCS is not meaningful, the *Best Common Subsumer* (BCS) has been defined [12].

3 Knowledge Compilation

`I.M.P.A.K.T.` takes all the information needed to model and manage the domain of human resources from a specifically developed modular ontology $\mathcal{T}$. The ontology currently includes nearly 5000 concepts modeling both the technical and the complementary competences an individual may hold.

In the following, we denote by $\mathcal{T} = \{M_i | 0 \leq i \leq 6\}$ the whole skills ontology adopted by the current implementation of `I.M.P.A.K.T.`. The ontology submodules M_i, with $i > 0$, are modeled according to $\mathcal{FL}_0(\mathrm{D})$ and describe different CV sections: `Industry`, `ComplementarySkill`, `Level`, `Knowledge`, `Language`, `JobTitle`. The ontology modularity allows for extending it whenever a new category of work-related features is identified, as shown below by the translation into a relational schema. Our default implementation of Core Competence extraction considers only `Knowledge` submodule, which models the hierarchy of possible candidate competence and technical tools usage ability; moreover, the module provides a `type` property to specify, for each competence, the related experience role (*e.g.*, developer, administrator, and so on) and two predicates: `year` to specify the experience level (in years), and `lastdate` which represents the last temporal update of work experience. M_0 is the main ontology module: it directly imports all the previous modules and models all of the properties needed for describing the candidate profile through the above detailed classes. We define as *entry points* such properties. In particular, M_0 includes one entry point for each imported sub-module.

Thanks to the knowledge modeling outlined so far, it is possible to model *CV Profiles* in the ABox. The CV classification approach we propose is based on a role-free ABox, which then includes only concept assertions of the form $P(a)$, stating that candidate a (*i.e.*, her CV description) offers profile features P (see Definition 2).

Definition 2. *Given the skill ontology $\mathcal{T}$, a **profile** P is a $\mathcal{ALE}(\mathrm{D})$ concept defined as a conjunction of existential quantifications, $P = \sqcap(\exists R_j^0.C)$, with $1 \leq j \leq 6$, where R_j^0 is an **entry point** and C is a concept in $\mathcal{FL}_0(\mathrm{D})$ modeled in M_j.*

As hinted before, our knowledge compilation problem aims at translating the skill knowledge base into a relational model, without loss of information and expressiveness w.r.t. our previous and fully logic-based work ([14]), in order to reduce on-line reasoning time. So, relational schema modeling is the most crucial design issue and it is strongly dependent on both knowledge expressiveness to be stored and reasoning to be provided over it. In particular, information storage involves both TBox axioms – concepts definitions, subsumption relations, value restrictions and profile descriptions – and ABox assertions – namely, profiles data represented as profile description instances – along with extra-ontological personal information. Notice that all non-standard reasoning services performed by `I.M.P.A.K.T.` process the atomic information making up the knowledge descriptions rather then the concept as a whole. For this reason, the availability of a finite normal form for such descriptions turns out to be very useful and effective. We recall that $\mathcal{FL}_0(\mathrm{D})$ concepts can be normalized according to the *Concept-Centered Normal Form* (CCNF), [7, Ch.2].

According to such an approach, the KB is mapped to the database by means of the following design rules: **1)** a table CONCEPT is created to store all the atomic information managed by the system; **2)** three tables mapping recursive relationships over the table CONCEPT– namely PARENT, ANCESTOR and DESCONCEPT– are created; **3)** a table PROFILE includes the profile identifier ($profileID$ attribute) and the so called *structured information*: extra-ontological content, such as personal data (***e.g.***, last and first name, birth date) and work-related information (***e.g.***, preferred working hours, car availability); **4)** a table $R_j(X)$ is created for each entry point R_j^0, where $X = \{profileID, groupID, conceptID, value, lastdate\}$.

Given that for each conjunct $\exists R_j^0.C$ in $P(a)$, `I.M.P.A.K.T.` adds one tuple for each atom of the $CCNF(C)$ to the table $R_j(X)$, the attribute `groupID` is needed to convey all the informative content (*i.e.*, atoms of $CCNF(C)$) referred to the same conjunct. Thus, all features modeled in profile descriptions (Definition 2) are stored in tables $R_j(X)$ related to the involved entry points.

The relational schema resulting from the rules detailed so far allows for flexible skill matching classes, automated team composition, logic-based ranking and explanation of results (see [8] and [9] for more details). Here, we present formally only one match class, namely *Strict Match*, because it is at the basis of the development of *Core Competence extraction* in `I.M.P.A.K.T.`.

Definition 3. *Given the ontology $\mathcal{T}$, a set $\mathcal{FS} = \{fs_1, \ldots, fs_s\}$ of required candidate features, with each fs_i of the form $\exists R_j^0.C_i$, and a set $\mathcal{P} = \{P(a_1), \ldots, P(a_n)\}$ of candidate profiles, modeled according to Definition 2 and stored in the DB according to the schema detailed so far, the* ***Strict Match*** *process returns all the candidate profiles in $\mathcal{P}$ providing all the features fs_i in $\mathcal{FS}$.*

We notice that, thanks to CCNF, *Strict Match* can retrieve candidate profiles $P(a)$ more specific than $\mathcal{FS}$. If we think of $P(a)$ and $\mathcal{FS}$ in terms of their corresponding $\mathcal{ALE}(\mathrm{D})$ description (according to Definition 2), we may assert that *Strict Match* retrieves all the provided candidate Profiles ($P(a)$) linked by a subsumption relation to a target competence ($\mathcal{FS}$). Once the knowledge base $\mathcal{K}$ has been pre-processed and stored into the DB according to our relational schema, `I.M.P.A.K.T.` is able to perform all the reasoning services –also the *Strict Match* – only through standard SQL queries (see [8] for more details). From database querying point of view, fs_i has to be translated in a set of syntactic elements to search for in the proper $R_j(X)$ table. Then, for each fs_i one query $Q_s(fs_i)$ is automatically built on-the-fly considering a number of conditions in `WHERE` clause defined according to atoms in C_i. The final result of *Strict Match* is the intersection of profiles returned by all performed $Q_s(fs_i)$ queries. In the following, it is shown an executable example for the request $fs_i = \exists hasknowledge.(Java \sqcap \forall skillType.Programming \sqcap \geq_3 years)$ (the query has three conditions in `WHERE` clause, as the reader may easily expect).

```
SELECT profileID FROM hasKnowledge as R
WHERE conceptID = (SELECT conceptID
                   FROM concept WHERE name='Java')
AND EXISTS (SELECT * FROM hasKnowledge
            WHERE profileid=R.profileid AND groupid=R.groupid
            AND conceptid = (SELECT conceptID
                           FROM concept
                           WHERE name='skillType.Programming'))
AND EXISTS (SELECT * FROM hasKnowledge
            WHERE conceptid=(SELECT conceptID
                           FROM concept WHERE name='years')
            AND value >= 3 AND profileid=R.profileid
            AND groupid=R.groupid)
```

4 Core Competence Extraction

As hinted before, our proposal to Core Competence Extraction takes the competence-based perspective, which interprets company strategic competence as a collective asset, resulting from the synergy of human resources.

We observe that, even though a fully (as concerns both representation and reasoning) logic-based Knowledge Management System solving Core Competence extraction have been presented in [14], the novelty of our proposal lies in adopting a knowledge compilation approach, making the computational cost of the whole process to be affordable also for large organizations, while retaining the full expressiveness of the logic-based approach. Therefore, coherently with the novel knowledge compilation approach introduced in this paper, we show in the following how to solve Core Competence extraction through the *Strict Match* execution over the database schema presented in Section 3. To this aim, `I.M.P.A.K.T.` services rely on the computation of partial Common Subsumers defined in Section 2.

In [12] the *Common Subsumer Enumeration* algorithm was proposed, determining the sets $\underline{BICS}$, $\underline{CS_k}$, $\underline{ICS_k}$, $\underline{BCS}$ of, respectively, BICS, k-CS, IkCS, BCS of a collection $\{C_1, \ldots, C_n\}$, given $k < n$. The algorithm extracts from the set of profiles at hand the knowledge components shared by a significant number of individuals in the set, with such a significance level to be set as a threshold value (k) by the people in charge for strategic analysis. The algorithm works by taking as input a concept collection in the form of a *Subsumers Matrix* and the threshold value k.

`I.M.P.A.K.T.` implements the above recalled algorithm. Nevertheless, in order to cope with the features of the concept collection at hand, we here need to redefine a *Profiles Subsumers Matrix* (see Definition 5), and its preliminary Definition 4. We notice that, according to Definition 2, a Profile embeds all of the qualitative information detailed in the company CVs. Instead, in the current implementation, the identification of company Core Competence is limited to the evaluation of technical knowledge. Such an assumption affects the following definition of Profile Concept Components, defined w.r.t. a set EP of entry points R_j^0 of interest.

Definition 4. *Let P be a profile according to Definition 2, $P = \sqcap(\exists R_j^0.C)$, with C in $\mathcal{FL}_0(\mathrm{D})$ written in a CCNF $C^1 \sqcap \ldots \sqcap C^m$ and let EP be a set $EP \subseteq \{1, ...6\}$. For $j \in EP$, the* ***Profile Concept Components*** *(PCC) of P w.r.t. EP are defined as follows:*
1. if C^k, with $1 \leq k \leq m$, is a concept name, then $\exists R_j^0.C^k$ is a PCC of P;
2. if C^k, with $1 \leq k \leq m$, is a concrete feature, then the concept $\exists R_j^0.(C^k \sqcap C^f)$ is a PCC of P, for each $f \in \{1, \ldots m\}$ such that $f \neq k$ and C^f is a concept name;
3. for each C^k, if $C^k = \forall R.E$, with $1 \leq k \leq m$, then, for each E^h PCC of E, the deriving PCC of P are: i) $\exists R_j^0.\forall R.E^h$; ii) $\exists R_j^0.(\forall R.E^h \sqcap C^f)$, for each $f \in \{1, \ldots m\}$ such that $f \neq k$ and C^f is a concept name.

The identification of profile concept components is at the basis of the *Profiles Subsumers Matrix* construction, defined as follows.

Definition 5. *Let $\mathcal{P} = \{P(a_1), \ldots, P(a_n)\}$ be the set of profiles modeled according to Definition 2. Let now EP be a set such that $EP \subseteq \{1, ...6\}$, and $D_k \in \{D_1, \ldots, D_m\}$ be the* PCC *w.r.t. EP deriving from the collection $\mathcal{P}$. Given Definition 3, we define the **Profiles Subsumers Matrix** (PSM) $S = (s_{ik})$, with $1 \leq i \leq n$ and $1 \leq k \leq m$, such that: (i) $s_{ik} = 1$ if $P(a_i)$ strictly matches the component D_k; (ii) $s_{ik} = 0$ if $P(a_i)$ does not strictly match the component D_k.*

The above introduced characterization of the set $\mathcal{P}$ of the available knowledge profiles allows *Common Subsumer Enumeration* algorithm to retrieve the set of common subsumers useful to determine company Core Competence. In order to improve readability of the rest of the paper, the following definitions are provided.

Definition 6. *Referring to the PSM of the set $\mathcal{P} = \{P(a_1), \ldots, P(a_n)\}$, we define:*
***Concept Component Signature** (sig_{D_k}) the set of indexes of profiles $\{P(a_1), \ldots, P(a_n)\}$ strictly matching D_k (note that $sig_{D_k} \subseteq \{1, \ldots, n\}$);*
***Concept Component Cardinality** (T_{D_k}) the cardinality of the set sig_{D_k}, that is, how many profiles in $\mathcal{P}$ strictly match D_k. Such a number is $\sum_{i=1}^{n} s_{ik}$.*

5 System Performances

Core Competence extraction has been implemented as an enterprise business service for the `I.M.P.A.K.T.` system. It is a client-server application developed in Java exploiting Jena API to access the ontology model and Pellet reasoner to classify the ontology in the off-line pre-processing phase. Current implementation uses the open source PostgreSQL 9.1 DBMS.

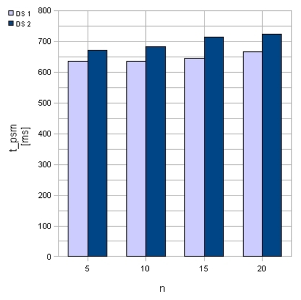
In order to prove the effectiveness and efficacy of `I.M.P.A.K.T.` services, we initially created a real data set by collecting about 180 CVs, from three different employment agencies of candidates specifically skilled in ICT domain, so to simulate the scenario of an actual company in the ICT industry. The same dataset has been exploited for an iterative refinement phase of both the Skill Ontology development and testing of results of `I.M.P.A.K.T.` services.

Here, we are interested in the evaluation of *data complexity* and *expressiveness complexity* of our knowledge compilation approach to Core Competence extraction. To this aim, we perform two different test campaigns: the first one compares the performance of our implementation w.r.t. a fully logic-based one ([14]); the second one is focused on showing scalability capabilities. In each test category we adopted a different pair of datasets (namely, DS_1 and DS_2 in the first and DS_3 and DS_4 in the second category). Items in each mentioned pair differ in the level of specificity of the included profiles: the first dataset (DS_1, or, respectively DS_3) is more generic than the second one (DS_2, or, respectively DS_4) and thus characterized – for the same number of profiles – by a smaller set of resulting profile concept components.

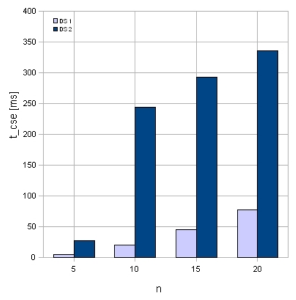
For all the performed tests, `I.M.P.A.K.T.` was executed on an Intel Dual Core server, equipped with a 2.26 GHz processor and 4 GB RAM. Moreover, only CV information related to technical knowledge have been considered for Core Competence extraction (i.e. concepts of the form $\exists R_j^0.C$, where the entry point $R_j^0 = hasKnowledge$).

We evaluate the two main extraction steps: the *Profile Subsumers Matrix* computation and the *Common Subsumer Enumeration* (CSE) algorithm execution (we recall that such algorithm takes the PSM as input). In the following, t_{psm} represents the average PSM computation time, t_{cse} represents the CSE time and n the number of profiles.

Figure 1 and Figure 2 show the performance results referring, for DS_1 and DS_2, to subsets of 5, 10, 15 and 20 profiles (characterized by the same cardinality as those in [14]). Such a different sets cardinality is aimed at investigating on how the execution time increases with the number of analyzed profiles. We also notice that, thanks to the adoption of the two datasets DS_1 and DS_2, it is possible to investigate on the relation between execution time and the number of profile concept components resulting from the dataset, when the number of profiles is given. Such a test strategy allows for assessing the impact of profiles complexity on the execution time. In particular, in Figure 1, we show t_{psm} (Figure 1(a)) and t_{cse} (Figure 1(b)), both in milliseconds, vs. n. Figure 2 presents the same computation times vs. the profile concept components number. Moreover, both Figure 2(a) and Figure 2(b) refer to the profile concept components deriving from DS_1 and DS_2. Intuitively, for each n value in Figure 2(a) and Figure 2(b), the smaller computation time value refers to DS_1 and the bigger to DS_2. In both experiments, k is set to 3.



(a)



(b)

Fig. 1. PSM (a) and CSE (b) computation time vs. number of profiles

It can be noticed that the number of profiles is highly relevant in the common subsumer enumeration process, while the profile subsumers matrix computation time is less affected by such a number. As a general remark, t_{psm} is bigger than t_{cse}: the matrix creation is the most computationally expensive process. Nevertheless, by comparing presented matrix computation times with the ones shown in [14], the reader can notice how the adopted knowledge compilation approach dramatically improves process performance, by taking execution time from seconds to milliseconds (as an example, matrix computation time for 20 profiles has changed from 180 seconds to about 660 milliseconds), paving the way to the use of the approach in large real-world scenarios.

For the second test campaign, aimed at evaluating the approach scalability, we carried out tests on pairs of synthetic datasets (subsets of the above mentioned DS_3 and

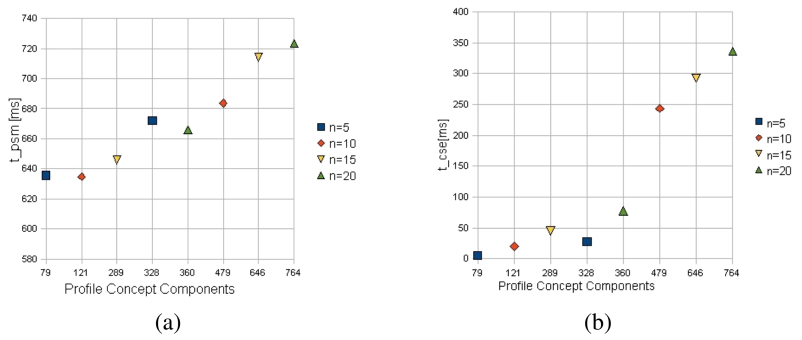


Fig. 2. PSM (a) and CSE(b) computation time vs. profile concept components

DS_4), such that the number of features for each candidate is comparable to the average value of candidate profile features in the training real dataset. In particular, we implemented a KB instances generator, able to automatically create satisfiable profiles, according to Definition 2, also setting the features format (*i.e.*, number of features for each entry point, minimum number of specified technical skills, and so on). Thanks to the generator, we randomly created one dataset DS_3 of 500 profiles and extended it to 1000 and 2000 profiles. Then, we created a different dataset DS_4 of 500 profiles, by specifying a bigger average number of technical skills to be generated (30 instead of 3), so that the resulting profile concept components number increases (in the first case components arise up to 11643 for 2000 profiles, while in the second to 28177). Such a dataset has been extended to 1000 and 2000 profiles, too. In Table 1, the execution time for the two main steps of Core Competence extraction processes are shown w.r.t. DS_3 and DS_4 and $k = 0.3 \times n$.

		Datasets Cardinality		
		500	1000	2000
t_{psm}	DS_3	0.87	1.1	1.74
	DS_4	3.91	9.24	27.29
t_{cse}	DS_3	0.21	0.46	1.4
	DS_4	66.06	235.81	912.57

Table 1. Core Competence extraction times (in seconds)

It can be noticed that, if we consider DS_3, the most computationally expensive process is still the Profile Subsumers Matrix creation. On the contrary, in presence of significantly complex profiles – and consequently of a huge number of deriving concept components – the Core Competence Enumeration execution time dramatically raises (see values of t_{cse} referred to DS_4 and Figure 2(b)). As a consequence, we may hypothesize that, for an increasing number of concept components, there is a critical value after which the most time-consuming phase switches from the matrix computation to the common subsumers sets identification.

In order to provide a further rationale of the Common Subsumer Enumeration algorithm, we now show its results w.r.t. a subset of 8 CVs (out of the 180), modeled according to Definition 2 and stored in the DB according to the schema detailed in Section 3. The profiles example set is given hereafter, with reference only to technical knowledge (*i.e.*, $R_j^0 = hasKnowledge$):

Mario Rossi: Cplusplus (5 years), Java (5 years), Visual Basic(5 years)

Daniela Bianchi: Cplusplus (2 years), Java (6 years), Visual Basic (1 years)

Lucio Battista: DBMS (2 years)

Mariangela Porro: DBMS (2 years), Internet Technologies (2 years)

Nicola Marco: DBMS (5 years), Internet Technologies (5 years)

Carmelo Piccolo: VBScript, Process Performance Monitoring

Elena Pomarico: CplusPlus, Java, Visual Basic

Domenico De Palo: OOprogramming (6 years), Artificial intelligence (4 years), Internet technologies (4 years)

Ontology concept inclusions needed for understanding the proposed example are shown in the following:

$\texttt{ComputerScienceSkill} \sqsubseteq \texttt{EngineeringAndTechnologies} \sqsubseteq \texttt{Knowledge}$
$\texttt{ProgrammingLanguage} \sqsubseteq \texttt{SWDevelopment} \sqsubseteq \texttt{ComputerScienceSkill}$
$\texttt{OOP} \sqsubseteq \texttt{ProgrammingLanguage}$
$\texttt{VBScript} \sqsubseteq \texttt{ScriptLanguage} \sqsubseteq \texttt{ProgrammingLanguage}$
$\texttt{Java} \sqsubseteq \texttt{OOP}, \texttt{C++} \sqsubseteq \texttt{OOP}, \texttt{VisualBasic} \sqsubseteq \texttt{OOP}$
$\texttt{MySQL} \sqsubseteq \texttt{RDBMS} \sqsubseteq \texttt{OpenSourceDBMS} \sqsubseteq \texttt{DBMS} \sqsubseteq \texttt{InformationSystem} \sqsubseteq \texttt{ComputerScienceSkill}$
$\texttt{InternetTechnologies} \sqsubseteq \texttt{ComputerScienceSkill}$
$\texttt{ArtificialIntelligence} \sqsubseteq \texttt{ComputerScienceSkill}$
$\texttt{ProcessPerformanceMonitoring} \sqsubseteq \texttt{ManagerialSkill} \sqsubseteq \texttt{BusinessManagement} \sqsubseteq \texttt{Knowledge}$

In table 2, we sketch the structure of the profile subsumer matrix with reference to a subset of profile concept components (explained in Table 3), determined from the set $\mathcal{P}$ of 8 CVs according to Definition 4 (actual matrix dimension is 8×89).

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	...
1	1	1	1	1	1	0	1	0	1	1	1	...
2	1	1	1	1	1	0	1	0	1	1	1	...
3	1	1	0	0	0	1	0	0	0	0	0	...
4	1	1	0	0	0	1	0	1	0	0	0	...
5	1	1	0	0	1	1	0	1	0	0	0	...
6	1	0	1	0	0	0	0	0	0	0	0	...
7	1	0	1	1	0	0	0	0	1	1	1	...
8	1	1	1	1	1	0	1	1	0	0	0	...

Table 2. Portion of the example Profile Subsumers Matrix

D_1	$\exists\texttt{hasKnowledge}.\texttt{ComputerScienceSkill}$
D_2	$\exists\texttt{hasKnowledge}.(\texttt{ComputerScienceSkill} \sqcap =_2 \texttt{years})$
D_3	$\exists\texttt{hasKnowledge}.\texttt{ProgrammingLanguage}$
D_4	$\exists\texttt{hasKnowledge}.\texttt{OOP}$
D_5	$\exists\texttt{hasKnowledge}.(\texttt{ComputerScienceSkill} \sqcap =_5 \texttt{years})$
D_6	$\exists\texttt{hasKnowledge}.(\texttt{DBMS} \sqcap =_2 \texttt{years})$
D_7	$\exists\texttt{hasKnowledge}.(\texttt{OOP} \sqcap =_5 \texttt{years})$
D_8	$\exists\texttt{hasKnowledge}.(\texttt{InternetTechnologies} \sqcap =_2 \texttt{years})$
D_9	$\exists\texttt{hasKnowledge}.\texttt{C++}$
D_{10}	$\exists\texttt{hasKnowledge}.\texttt{VisualBasic}$
D_{11}	$\exists\texttt{hasKnowledge}.\texttt{Java}$
...	

Table 3. Description of $D_1, \ldots, D_{11}$ reported in Table 2

According to the process introduced in Section 4, the system implements CSE algorithm and searches for profile concept components D_j whose cardinality T_{D_j} is greater or equal than a predefined threshold value k. In particular, k is set to the total number of profiles in the data set for the LCS computation. For the example set, the LCS computation returns: $\underline{LCS} = \{\exists \texttt{hasKnowledge}.\texttt{ComputerScienceSkill}\}$, which is quite a generic result for an ICT company.

The algorithm reveals also a slightly more significant information: the common know-how shared by the biggest portion of company personnel (but not by all of it) is: $\underline{BICS} = \{\exists \texttt{hasKnowledge}.\texttt{ComputerScienceSkill} \sqcap =_5 \texttt{years}\}$.

Finally, the process reveals that, for a degree of coverage set to 3, the set of company Core Competence includes all the following elements: $\underline{ICS}_3 = \{\exists \texttt{hasKnowledge}.(\texttt{DBMS} \sqcap =_2 \texttt{years}), \exists \texttt{hasKnowledge}.(\texttt{OOP} \sqcap =_5 \texttt{years}), \exists \texttt{hasKnowledge}.(\texttt{InternetTechnologies} \sqcap =_2 \texttt{years}), \exists \texttt{hasKnowledge}.(\texttt{C++} \sqcap \texttt{VisualBasic} \sqcap \texttt{Java})\}$.

In Figure 3, the `I.M.P.A.K.T.` Graphical User Interface (GUI) for the Core Competence extraction process is shown. Panel (a) provides the input user interface for choosing the degree of coverage k and the desired entry points to be considered in the extraction process. Panel (b) lists all possible pieces of company Core Competence, providing a user with the possibility to visualize (in panel (c)) the personnel holding such a strategic asset.

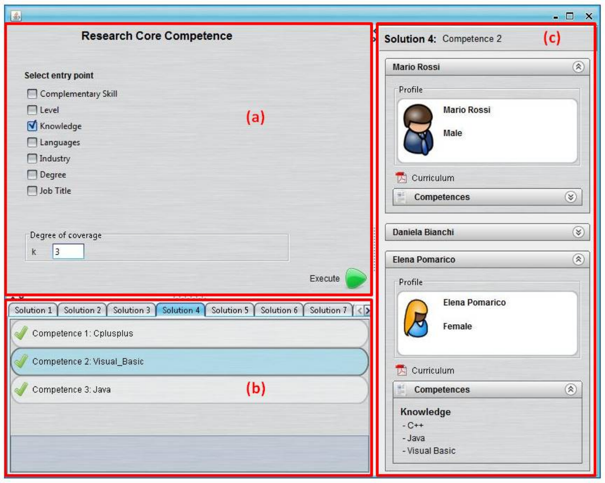


Fig. 3. `I.M.P.A.K.T.` GUI for Core Competence extraction

6 Conclusions

In the framework of `I.M.P.A.K.T.`, an integrated system providing several knowledge management services, we showed how the process of company Core Competence extraction can be significantly improved in terms of performance by adopting a Knowledge Compilation approach. Thanks to such an approach, the full informative content modeled in a DL knowledge base has been translated in a relational database schema where advanced inference services can be executed exploiting standard-SQL queries. We showed, through experimental evaluation, the dramatic reduction obtained in terms of execution times with our novel approach. Implementation of database modeling denormalization and table partitioning are under way, and should further improve the performances.

Acknowledgements

We acknowledge support of projects UE ETCP "G.A.I.A." and Italian PON "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems".

References

1. Draganidis, F., Mentzas, G.: Competency based management: A review of systems and approaches. Inform. Manag. and Computer Security **14**(1) (2006) 51–64
2. Cadoli, M., Donini, F.M.: A survey on knowledge compilation. AI Commun. **10**(3-4) (1997) 137–150
3. Hamel, G., Prahalad, C.K.: The core competence of the corporation. Harvard Business Review **May-June** (1990) 79–90
4. Hafeez, K., Zhang, Y., Malak, N.: Core competence for sustainable competitive advantage: a structured methodology for identifying core competence. IEEE Transactions on Engineering Management **49**(1) (2002) 28–35
5. Tampoe, M.: Exploiting the core competences of your organization. Long Range Planning **27**(4) (1994) 66 – 77
6. Sanchez, R., Heene, A.: Reinventing strategic management: New theory and practice for competence-based competition. European Management Journal **15**(3) (1997) 303 – 317
7. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook – 2nd edition. Cambridge University Press (2007)
8. Tinelli, E., Colucci, S., Giannini, S., Sciascio, E.D., Donini, F.M.: Large scale skill matching through knowledge compilation. In: Proc. of ISMIS 2012, Springer-Verlag (2012) 192–201
9. Tinelli, E., Colucci, S., Di Sciascio, E., Donini, F.M.: Knowledge compilation for automated team composition exploiting standard SQL. In: Proc. of SAC 2012, ACM (2012) 1680–1685
10. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Efficient Reasoning on Large SHIN Aboxes in Relational Databases. In: SSWS '09, CEUR (2009) 110–124
11. Kiryakov, A., Ognyanov, D., Manov, D.: Owlim - a pragmatic semantic repository for owl. In: Proc. of WISE'05. Volume 3807., Springer (2005) 182–192
12. Colucci, S., Di Sciascio, E., Donini, F.M.: A knowledge based solution for core competence evaluation in human capital intensive companies. In: Proc. of I-KNOW'08, Springer (2008) 259–266
13. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Proc. of AAAI'92, AAAI Press (1993) 754–760
14. Colucci, S., Tinelli, E., Sciascio, E.D., Donini, F.M.: Automating competence management through non-standard reasoning. Engineering Applications of Artificial Intelligence **24**(8) (2011) 1368 – 1384