The Tomaco Hybrid Matching Framework for SAWSDL Semantic Web Services

Thanos G. Stavropoulos, Stelios Andreadis, Nick Bassiliades, *Member, IEEE,*
Dimitris Vrakas and Ioannis Vlahavas

Abstract— This work aims to advance Web Service retrieval, also known as Matching, in two directions. Firstly, it introduces a matching algorithm for SAWSDL, which adapts and extends known concepts with novel strategies. Effective logic-based and syntactic strategies are introduced and combined in a novel hybrid strategy, targeting an envisioned well-defined, real-world scenario for matching. The algorithm is evaluated in a universal environment for matching algorithms, SME2, in an objective, reproducible manner. Evaluation ranks Tomaco high amongst state of the art, especially for early recall levels (first in macro-averaging precision, up to 0.7 recall). Secondly, this work introduces the Tomaco web application, which aims to promote wide-spread adoption of Semantic Web Services while targeting the lack of user-friendly applications in this field, by integrating a variety of configurable matching algorithms proposed in this paper. It, finally, allows discovery of both existing and user-contributed service collections and ontologies, serving also as a service registry.

Index Terms— Web Services Discovery, Intelligent Web Services and Semantic Web, Internet reasoning services, Web-based services

1 Introduction

THE Web currently enables billions of users to look up massive amounts of data and carry out transactions, playing a vibrant role in everyday lives. Since its creation, it has evolved from hosting plain data on static pages to dynamic crowd-sourcing content publication through blogging and social networks. On top of data retrieval, web users can now perform actions and carry out tasks enabled by the Web Service technology. Web Services entail a set of technologies and standards to properly define APIs of remote procedure calls (RPCs). Since Web Services need to be discovered before they are invoked, the Semantic Web [1] technologies have been employed to aid that cause. Semantics on service descriptions render them machine interpretable and thus enhance Discovery [2], Selection, Matching and Composition [3].

The Semantic Web Service concepts have emerged from the synergy of Web Service and Semantic Web technologies. Early works in the field followed a top-down approach of describing services using high-level ontological constructs, such as the OWL-S upper ontology for services [4] or the WSMO Web Service Modeling Language [5]. These so-called upper ontologies define hierarchies of concepts for modeling and describing a service, its workflow, grounding and, most importantly, the Input, Output, Preconditions and Effects (IOPEs) of its operations. Ontological descriptions provide much flexibility and expressiveness, but are less strict towards bindings to actually invoke the Services in question, since WSDL groundings are optional. Additionally, their complexity and subjective interpretation has hindered their wide adoption by the industry. This led to the emergence of lightweight, bottom-up approaches, such as SAWSDL [6] and WSMO-lite [7], which provide compact annotations on WSDL groundings themselves. Amongst them, SAWSDL became a W3C recommendation and a leading Semantic Web Service description methodology, which is increasingly adopted in industry and academia. Despite its low complexity and expressiveness, it has already been proven to be suitable for enhancing Web Service Discovery [2], Selection, Matching and Composition [3] [8]. Yet, these notions are just starting to penetrate the real-world with applications, e.g. [9].

This work serves a two-fold contribution towards accessible Semantic Service Matching i.e. the problem of user or software-driven search for suitable services, or in essence service operations, using structured criteria i.e. input and/or output, in accordance to SAWSDL capabilities (which exclude e.g. effects). The first contribution is a proposed Semantic Web Service matching strategy tailored to the SAWSDL lightweight schema. After a thorough review of state-of-the-art algorithms that target SAWSDL and some interesting work on OWL-S/WSMO, most strategies can be classified in categories, such as semantic (logic-based), syntactic (IR-based), structural, learning and hybrid. The proposed algorithm aims to adopt, adapt and combine some of these elements, to target an envisioned use case scenario of well-defined requirements under which it performs optimally, without excluding different circumstances. In this scenario, the user requires most relevant results early (top of the list) and fast response time,

- *T. G. Stavropoulos is with the Aristotle University of Thessaloniki, Greece. E-mail: athstavr@csd.auth.gr.*
- *S. Andreadis is with the Aristotle University of Thessaloniki, Greece. E-mail: sandreai@csd.auth.gr.*
- *N. Bassiliades is with the Aristotle University of Thessaloniki, Greece. E-mail: nbassili@csd.auth.gr.*
- *D. Vrakas is with the Aristotle University of Thessaloniki, Greece. E-mail: dvrakas@csd.auth.gr.*
- *I. Vlahavas is with the Aristotle University of Thessaloniki, Greece. E-mail: vlahavas@csd.auth.gr.*

over a dynamic service registry.

The hybrid technique introduced in this work employs a novel logic-based strategy, complemented by text-similarity measures on both semantics and textual descriptions. Evaluation on a large, publicly available dataset has proven that the logic-based technique has the greatest impact on retrieving relevant services. These findings are inline with works such as [10], where feature vectors extracted from semantic information for retrieval learning were found more effective than those originating from syntactic information. However, although text-similarity methods give poor results on their own, they do compensate for losses of the logic-based strategy in this work. They also enable semantic matching of non-annotated (plain WSDL) descriptions. Various measures e.g. macro-averaging precision at standard recall levels, F1-score, and average precision (AP) are used to appreciate the effectiveness of each variant. The proposed method has ranked high amongst state-of-the-art in both effectiveness and performance.

The second contribution of this work is the integration of the proposed matching strategies in the user-oriented Tomaco web application (Tool for Matching and Composition). Tomaco is a publicly available web application[1] that aims to render Semantic Web Service exploitation easily accessible to experts and non-experts alike. Apart from invoking strategies with a variety of parameters to choose from, users are able to target existing service collections or upload their own. This way, Tomaco also serves as a service registry, allowing discovery using semantic criteria.

This paper is structured as follows: the next section surveys existing state-of-the-art algorithms, datasets and registries, identifying the main principles of matchmaking applications. The third section introduces the Tomaco algorithm, describing its four matching strategies, implementation and extensive evaluation runs. The fourth section presents the Tomaco web application, its architecture and functionality. Future directions and conclusions drawn are presented on the corresponding final section.

2 Related Work

This section lists indicative examples of state-of-the-art in matching algorithms and service registries, while highlighting comparisons to the proposed system.

2.1 Web Service Matching Algorithms

Existing techniques, as presented on Table 1 and described in the following subsections, can be classified as logic-based, syntactic or text-similarity-based, while structural similarity and learning are less common. Logic-based techniques range from straightforward series of few class-relationship rules to large lists of semantic conditions. Notice that, despite the name, this family of techniques mainly employs reasoning upon class subsumption and equivalence, avoiding more complicated inferences. Text-similarity mostly uses textbook algorithms from the field of Information Retrieval. Each existing technique is ultimately compared to the one proposed in this paper. Measures of effectiveness for most of them are also presented in the evaluation section. When describing logic-based techniques, offered (or provided) and requested services from now on will be denoted as O and R respectively. Input and output are denoted in subscript; e.g. R_i stands for requested input. Superclass and subclass relationships of a left-hand side entity to the right-hand side are denoted using > and < respectively.

SAWSDL-MX, *SAWSDL-TC*, *iSeM*

The work in [11] presents two major contributions. First of all, the authors provided the first and largest dataset of more than a thousand SAWSDL files, namely SAWSDL-TC (Test Collection). To do so, they used expert manual labor combined with the OWLS2WSDL tool to map OWL-S descriptions (from OWLS-TC2.2) to lightweight format. The test collection's updated version, SAWSDL-TC3, found online[2], contains 1080 service documents, OWL ontologies, queries and relevant sets. Secondly, the authors present an initial approach to exploit this test collection towards automatic matching. Following the dataset's transformation, SAWSDL-MX is an adaptation of previous works OWLS-MX [12] and WSMO-MX [13].

The algorithm provides all standard matching strategies, namely logic-based, syntactic (text-similarity) and hybrid (logic-based and syntactic similarity). The strategies target service input, output and their underlying components (e.g. ComplexType), trying to find a match between a requested service (i.e. query) and all services offered in a set. Each offered service's operation is matched with every requested operation and rated with the maximum observed match. An offered service's overall rating is the worst (minimum) rating of all requested operations. However, SAWSDL-TC contains single-operation services only. Hence, operation-match rating is equal to the overall service-match rating.

Rating scores for the logic-based strategy are set according to the following order from highest to lowest:

1. *Exact:* perfect matching of inputs and outputs i.e. $R_i = O_i \wedge R_o = O_o$
2. *Plug-in:* offered input is arbitrarily more general than requested input and offered output is a direct child of requested output i.e. $O_i > R_i \wedge O_o <_{direct} R_o$
3. *Subsumes*: inputs as in *Plug-in* and offered output is a(ny) child of requested output (relaxes output constraints) i.e. $O_i > R_i \wedge O_o < R_o$
4. *Subsumed-by:* inputs as in *Plug-in* and offered output is a direct parent of requested outputs i.e. $O_i > R_i \wedge O_o >_{direct} R_o$
5. *Fail:* none of the above applies

When multiple annotations are present, the strategy

[1] Tomaco web application: http://tomaco.csd.auth.gr
Tomaco Homepage: http://lpis.csd.auth.gr/people/thanosgstavr/applications/tomaco.html

[2] SAWSDL-TC http://projects.semwebcentral.org/projects/sawsdl-tc/

considers the first one only, while input and output match ratings count equally for the overall service rating. The syntactic strategy applies IR (Information Retrieval) methods i.e. various text-similarity algorithms, provided by the SimPack[3] Java library (e.g. Loss-of-information, extended Jaccard, Cosine and Jensen-Shannon), between requested and offered semantic elements. Finally, the hybrid strategy offers two variants. The *compensative variant* lets syntactic-similarity "compensate" when logic-based returns *Fail.* In the *integrative variant, Subsumed-by* matches are further constrained by a text-similarity-above-threshold requirement, and is, thus, more strict than logic-based. The algorithm is integrated along with a Service Registry, Ontology Handlers (which locate ontologies referred in services and host reasoning utilities) and an Ontology Registry. SAWSDL-MX has been evaluated over SAWSDL-TC, indicating the hybrid method (with Cosine text-similarity) as the most effective but also the slowest, followed by the syntactic and logic-based methods.

SAWSDL-MX2 [14], in addition to logic-based and text-similarity, measures structural similarity between WSDL file schema information (e.g. element names, data types and structural properties), using WSDL-Analyzer[4]. It also introduces an adaptive, learning layer where SVM training vectors consist of values for logic-based, textual and structural criteria and binary relevance: {*Exact, Plug-in, Subsumes, Subsumed-by, Fail,* text-similarity, structural-similarity, relevance}. Logic and structural similarity (M0 + WA), adaptive (MX2) and logic + textual hybrid (MX1) show no significant difference on AP, while improving over plain methods. However, M0 + WA and MX2 require double per query response time than MX1. These findings support our decision to employ hybrid logic + textual methods in Tomaco.

iSeM [15] is an evolution of the MX series by the same authors. In principle, it applies SVM learning for the weighted aggregation of underlying algorithm rankings. The learning vectors are an extended version of the ones in SAWSDL-MX2, containing logic, structural and text-similarity in similar fashion to SAWSDL-MX algorithms. However, approximate logic matching was added, which captures more matches than the existing one, using looser criteria for subsumption. The algorithm is extremely precise in SAWSDL-TC and has ranked first in AP, as measured in S3 (Semantic Service Selection Contest)[5] 2010 and 2012, with an expected trade-off in performance.

On the other hand, Tomaco adopts and extends selected aspects of the above works, contributing certain improvements. To begin with, the logic-based rating in SAWSDL–MX/iSeM shows much room for improvement. Tomaco's logic-based strategy considers more relaxed (less) rating criteria, e.g. by independently examining input and output. Also, Tomaco is operation-centric i.e. rates individual operations, as opposed to picking the lowest operation rating for a service (SAWSDL-MX). These "looser" rating criteria are carved out more via a use case scenario in the next section, while improvements in effectiveness are shown in the evaluation section. Syntactic techniques, also widely used in state-of-the-art, are adapted in Tomaco by using two different text-similarity algorithms which were found well-suited for web service element names and annotations, improving effectiveness. Pure methods are combined into an adapted hybrid strategy, again driven by fundamental principles carved out in the next section. In summary, our method is mostly targeted at web service retrieval use cases where high precision is needed at early recall levels, without excluding of course other situations.

Learning can also be used in web service matching, given a data and relevant set, in order to adaptively select the best strategy for a given instance or to tweak the configuration parameters of the various strategies. Indeed, it has been successfully employed in many works as presented in this and the evaluation section. However, since the only reliable dataset for learning is currently SAWSDL-TC, the model that will be learned cannot necessarily be generalized nor validated in another domain. For this reason, we have refrained from using learning in this work, without excluding this option for the future e.g. by forging extended datasets. Instead, Tomaco aims to provide an effective heuristic, general-purpose method to target services in the open world, without the need to re-train models as services come.

LOG4SWS, COV4SWS

XAM4SWS is a common framework, which derived two algorithms, LOG4SWS and COV4SWS [16]. Both algorithms perform operation-centric matching, targeting service interfaces, operations and I/O. LOG4SWS performs logic-based matching, in an SAWSDL–MX fashion, mapping ratings to numbers using linear regression. Meanwhile, COV4SWS rating measures are inspired from the field of semantic relatedness. It then performs regression to find weights for the aggregation of ratings (from underlying service elements to an overall service rating). Both algorithms fallback to WordNet similarity (inverse distance), if semantics are entirely absent. Both methods are highly effective on TC3, ranking first in nDCG (normalized Discounted Cumulative Gain for graded relevance) and Q-measure [17], while maintaining a fast response time. Tomaco, on the other hand, follows a different, heuristic approach, as previously discussed.

iMatcher

iMatcher [18] integrates interesting variations of well-known strategies. The first strategy includes three sub-strategies. It performs text-similarity (using Java SimPack) targeting either the WSDL service name field, service description field or semantic annotations. The second strategy selects the maximum rating between two sub-strategies. The first is a hybrid variant where the logic-based part rates inputs and outputs of operations with 1, if the requested concept is a parent of the offered concept ($R > O$). Hence, iMatcher's logic substantially differs from

[3] SimPack: http://www.research-projects.uzh.ch/p8227.htm
[4] WSDL-Analyzer: http://www.wsdl-analyzer.com/
[5] S3 Contest - http://www-ags.dfki.uni-sb.de/~klusch/s3/

SAWSDL-MX, which requires $R_o > O_o$ in *Plug-In* and *Subsumes* and the opposite, $R_i < O_i$, in all cases (*Exact, Plug-In* etc.). If logic-based matching fails, syntactic matching is performed (in the spirit of the compensative variant). The second sub-strategy examines distance of two concepts originating from different ontologies using ontology alignment similarity as obtained from the Lily tool[6]. On top of that, iMatcher also implements an Adaptive Matching method. The user selects multiple strategies, the results of which form vectors of the training set. Learning is performed by selecting an algorithm from the Weka library. Regarding effectiveness, iMatcher's best strategies are Adaptive Matching with Logistic and e-SVR learning, followed by hybrid with ontology alignment. Tomaco tries to improve upon those ideas by proposing a more effective and less strict logic-based technique which we believe accounts for an improved hybrid method.

Skyline

The Skyline system [19] performs matching on OWL-S descriptions instead of SAWSDL, but its interesting strategy is worth mentioning. The strategy's target components are IOPEs, grouping Inputs together with Preconditions and Outputs with Effects. First, it performs logic-based classification to *Exact*, *Direct_Subclass*, *Subclass*, *Direct_Superclass*, *Superclass*, *Sibling* and *Fail* (selecting the best match). The homonymous Skyline algorithm is used to find the optimal trade-off of input versus output significance. E.g. a service of *Exact* input and *Fail* output and a second of *Subclass* input and *Direct_Superclass* output will prevail over a third service of *Direct_Subclass* input and *Fail* output. Ratings for multiple R_i, R_o are also supported through the Skyline algorithm. Users are able to request the next skylayer from the algorithm to get more services. Skyline (0.83 AP) has ranked well above OWLS-MX (0.71 AP) on OWLS-TC2 dataset.

The Skyline technique has actually proven to be effective in the document retrieval domain. However, to adapt it in service matching, the authors had to consider semantic relationships as ordinal values, e.g. a superclass is worse than a subclass. Our approach does consider that some relationships are superior to others but this case differs for input and output concepts. Overall, the Skyline technique does seem interesting, if properly adapted, and may be investigated as future work.

Table 1. State-of-the-art comparison

System	Year	Format	Logic	Syntactic	Hybrid	Other
SAWSDL-MX	2008	SAWSDL	yes	yes	yes	-
SAWSDL-MX2	2009	SAWSDL	yes	yes	yes	Learn. Struct.
iSeM	2010	OWL-S, SAWSDL	yes	yes	yes	Learn. Struct.
LOG/COM 4SWS	2010	SAWSDL	yes	yes	yes	Learn.
iMatcher	2011	SAWSDL	yes	yes	yes	Learn.
Skyline	2008	OWL-S	yes	-	-	-
HSW	2009	SAWSDL	yes	yes	-	-
OOM	2007	OWL-S	yes	-	-	-
OWLS-SLR	2010	OWL-S	yes	-	-	-
IRS-III	2008	WSMO	yes	-	-	-
Themis-S	2010	WSDL	-	-	-	Lingual
URBE	2009	SAWSDL	yes	-	yes	Lingual
Tomaco	2015	SAWSDL	yes	yes	yes	-

Hybrid Strategy with WordNet

The SAWSDL matching system in [20], denoted by HSW, proposes a complex logic-based algorithm to classify Input and Output as *Precise*, *Over*, *Partial*, *Mismatch* according to different ratios of provided, matched and requested I/O. The algorithm entails a long series of rules for the classification (which arguably makes its practical meaning hard to grasp). E.g. when semantics are missing from the examined node but the parent node's semantics match, text-similarity and WordNet [21] distance on the examined node are used as a measure. No evaluation was performed to assess the system's effectiveness.

While WordNet seems to be a popular choice regarding similarity measures, we feel that the presented algorithm's criteria are too strict i.e. the logic-based strategy entails a complex series of conditions to be met for matching. The text-based and WordNet-based approaches also entail hard-to-meet criteria (i.e. missing semantics on the current node but similar semantics on the parent node), which does not allow them to compensate for the logic-based strategy. For that purpose, we propose a compensative hybrid technique to handle mismatches that indeed improves over pure strategies. Furthermore, WordNet is a lexicon itself, while service requests always come with their own lexicons, i.e. ontologies to serve as heuristics. However, it can be used as a semantic and syntactic measure all-in-one in a future endeavor.

Object-Oriented Measures and OWLS-SLR

The work in [22], denoted as OOM, proposes a novel method for measuring semantic service similarity. Simple Property (datatype) similarity considers: *exact match*, *numerical-type match* (e.g. xsd:int and xsd:float) and *mismatch*. Relational Property (logic-based) similarity returns the distance of two concepts in a hierarchy, or through a common ancestor. If no common ancestor exists or classes are disjoint, their distance is infinite and, thus, similarity is zero. The total rating is the product of Simple and Relational

[6] Lily tool : http://ontomappinglab.googlepages.com/oaei2007

scores. OOM evolved in OWLS-SLR [23], which considers relationships of both subsumption and siblings. It targets OWL-S signatures, looking into properties (roles) and classes. A merit of OWLS-SLR is its low response time, as it quickly finds an initial set of candidate descriptions. In comparison, Tomaco does not consider such advanced reasoning capabilities, but rather a straightforward logic-based and syntactic hybrid.

IRS-III

IRS-III [24] is an integral system for creation, execution and selection of WSMO and OWL-S descriptions. A custom ontology representation, OCML, encodes service descriptions. Goal Mediator matches requested capabilities, i.e. input types, preconditions and assumptions to non-functional properties. Mismatches occur when requested (Goal) inputs are different or fewer in number than the ones offered i.e. $R_i \neq O_i \vee |R_i| < |O_i|$. A more recent version of IRS-III took part in the S3 2009, ranking low in JGD (Jena Geography Dataset) experiments on both effectiveness (AP=0.41) and performance (2.826s per query). IRS-III's logic-based strategy is unclear and is hard to investigate. Additionally, using predefined fields and criteria in descriptions, e.g. assumptions, and non-functional properties, is considered a non-universal practice.

Themis-S

Themis-S [25] focuses on syntactic matching, applying no logic-based method. The so-called enhanced Topic-Based Vector Space Model (eTVSM), a variant of classic TVSM, is extracted from WSDL. It uses WordNet to find linguistic relations and rate cases as *synonymy*, *homonymy*, *hyponymy* or *hypernymy*. Themis-S is evaluated over a custom dataset from programmableweb.com[7] and seekda.com[8], showing optimistic results, and in the S3 contest. Although WordNet for text-similarity is a promising method, our work mainly demonstrates the superiority and effectiveness of hybrid logic + textual methods.

URBE

The work in [26] proposes the URBE/URBE-S system, which incorporates a hybrid SAWSDL matching algorithm. The logic-based strategy, named URBE-S, calculates *annSim* (annotation similarity) as the distance of two concepts in the same ontology (as in [22]). If semantics are non-existent (pure URBE case), *nameSim* finds linguistic similarity, targeting service name, operation name and I/O, using a domain-specific or general purpose ontology, such as WordNet in this implementation. In both cases, *DataTypeSim* calculates data type similarity between xsd:types in WSDL simpleTypes (only) according to a predefined table. Overall rating is set to the average of R_i, R_o ratings. Macro-averaging precision-recall diagram ranked URBE-S above plain URBE and various state-of-the-art algorithms in [26]. All in all, the URBE/URBE-S system, despite its long response time, justifies the effectiveness of semantics in matching algorithms.

2.2 Web Service Registries

Some past works have been more focused on providing service registries rather than effective matching algorithms. OPOSSum [27] is such a web-based registry of services, that also hosts large datasets e.g. JGD and OWLS-TC. However, the underlying retrieval technique is a keyword-based mapping to SQL queries and not based on semantics. A similar, but much more extensive effort can be seen in BioCatalogue [9], a large registry of services related to life science. BioCatalogue similarly does not support semantic queries, but extends keyword search with much more metadata search fields and a tag-cloud. Another similar example is WESS [28], a keyword-search service registry that discovers WSDL/SAWSDL and OWL-S files after targeted crawling over the Web. Finally, iServe [29] follows a different service description approach, using RDF/Linked Data to publish, analyze and discover services. Compared to such works, we do provide a much more concise service registry, but focus on semantic search and matching. In other words, Tomaco provides an algorithm for individuals to experiment with semantic matching algorithms and parameters on existing or user-provided service collections.

3 Tomaco Matching Algorithm

The Tomaco matching algorithm design was driven by a set of requirements, which aim to describe the target problem of service matching in an open-ended, dynamic registry. These requirements describe an ideal use case scenario for which the algorithm performs optimally, without excluding its use under different circumstances:

1. Operation-centric search: the user is looking for one or more suitable service operations
2. I/O: user queries provide semantic input and/or output search criteria independently
3. The most desired service equally matches search criteria. A highly desired service offers more general input and/or less general output than requested. A less desired service offers less general input and/or more general output than requested. A non-desired service offers none of the above
4. Services can be dynamically added by web users and queried equally effectively
5. Response time must be fast enough for the context of an end-user web application
6. Users are more interested in getting several relevant services within the top results, rather than finding all of them, in the potentially huge answer space

The requirements are thoroughly addressed throughout the four proposed strategies. In detail, all strategies rank operations (1), consider input and output independently (2), regardless of annotation depth and with certain logic-based criteria (3). The methods do not require adaptation and maintain a low response time (4, 5) while

[7] Repository of Web APIs: http://www.programmableweb.com/

[8] Repository of bookings: https://www.seekda.com/

ensuring high precision in low recall levels i.e. greater than 0.7 precision up to recall level 0.7 (6).

Two additional principles of technical nature drove strategy design further: 1) Web service element names, annotations and ontological concepts are often written according to programming practices, such as CamelCase or snake_case. Incidentally, some concepts are expressed as such in SAWSDL-TC (18% and 36% respectively). 2) Textual descriptions of ontological concepts in the form of free-text should not necessarily be provided in such open-world scenarios. Therefore, they are not currently considered in Tomaco. They can be rather open-ended and require a separate study of proper heuristics.

LOGIC-BASED STRATEGY

The proposed logic-based strategy collects semantic annotations from target components of the XML-based tree structure of SAWSDL/WSDL. Algorithm 1 shows the outline of the complete rating strategy. Instead of providing pseudocode for the extraction procedure (invoked at *op.getInput*, line 4 and *op.getOutput*, line 12), Fig 1 shows the standard WSDL structure to be explored in a recursive DFS manner, while collecting *sawsdl:modelReferences* when present. The algorithm begins from *interface* and goes down to tree leaves (usually *simpleTypes*) exhaustively (choosing any branch first, e.g. between input or output, type or element). Specifically, the unique service *interface* may have one or more *operations,* which in turn, have *input* and *output. Input* and *output,* through *message*, have *parts* or *elements.* Each *part* can contain a *type* or, again an *element. Types* can be either *simpleTypes* or *complexTypes.* The latter can contain a sequence *(xsd:sequence)* of *elements* and so it can infinitely decompose to simpler types. In all proposed strategies, we are aiming to loosely comprehend (and match) the semantics behind the wording of each element. Therefore, depth is not considered in our work, since: 1) different XML structures can express the same meaning and 2) this simplifies and relaxes rating criteria. However, depth information, easily obtained in Tomaco during DFS, can be investigated in future studies as to how and when they can improve precision.

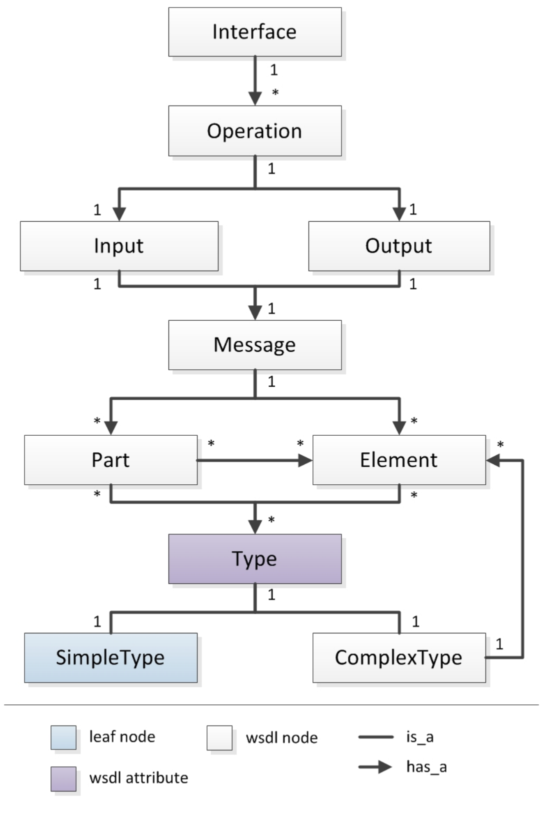


Fig 1. WSDL tree definition that guides DFS-traversal for the extraction of semantics

The logic-based strategy of rating semantics themselves handles characteristics of inputs and outputs differently, as presented in Algorithm 2. More specifically, it is based on a practical principle that, in the context of I/O-driven search, users may possess certain input information and/or desire certain output information. Hence, they can settle for more abstract input data, i.e. input offered is a superclass of required ($O_i > R_i$), but more specific input data ($O_i < R_i$) is less desired. The opposite of this principle applies for service outputs. The user desires to obtain certain output information from a service. If this information is more generic ($O_o > R_o$), it is of lesser use to the user, while more specific information ($O_o < R_o$) is of greater use. Some examples within the SAWSDL-TC dataset that conform to these principles are Genre > Science_Fiction (books.owl), MedicalOrganization > Hospital (HealthInsuranceOntology.owl) for input and City < UrbanArea (travel.owl), OpticalZoom < Zoom (extendedCamera.owl) in desired outputs.

Overall, the algorithm is driven by intuitively identifying a user's *desires.* We consider four matching cases: *Exact, Desired, LessDesired* and *Fail.* The *Exact* match is the most desired one, in both cases of input and output, and should be rated with maximum similarity. To rate the rest of the cases, two parameters are defined, an *UpperRate* and a *LowerRate*. Hence, when a *Desired* concept is offered, i.e. a superclass of input or a subclass of output, logic-based similarity is set to *UpperRate*. Likewise, if a *LessDesired* concept is found, i.e. a subclass of input or a superclass of output, rating is set to *LowerRate*. If no semantics are present or the concepts share no hierarchical relationship, the match is classified as *Fail.* The proposed method always handles similarity as a numerical value in [0, 1] in order to allow continuous values of ratings and facilitate combinations with other methods (e.g. hybrid). Through internal experiments, we have found the optimal values to be 0.75 for *UpperRate* (close but less than *Exact*) and 0.25 for *LowerRate* (close but higher than *Fail*).

- *Exact* : 1
- *Desired* : *UpperRate* = 0.75
- *LessDesired* : *LowerRate* = 0.25
- *Fail* : 0

The strategy entails a series of getting max values, average of vectors and applying weights between input and output, as shown on Algorithm 1. In detail, matching is performed by rating each offered operation. Two vectors are formed, one for requested inputs (line 5) and one for

Algorithm 1 Tomaco Matching

```
Input:
$R_i$: Array of requested input;
$R_o$: Array of requested output;
$O_s$: Set of offered services;
$w$: weight of input vs output significance, $\in (0,1)$
Output: Ranking of offered operations

1:  Array $OperationRatings \leftarrow \emptyset$;
2:  for each service $s \in O_s$
3:    for each operation $op \in s$.getOperations()
4:      Array $O_i \leftarrow op$.getInput().getSemantics();
5:      Array $ratingO_iR_i = \emptyset$      //max per $r_i$ ratings
6:      for each requested_input $r_i \in R_i$
7:        Array $ratingO_ir_i \leftarrow$ LogicBased($r_i$, $O_i$,'input');
8:        $ratingO_iR_i$.add($ratingO_ir_i$.getMax());
9:      end for
10:     double $ratingO_i \leftarrow ratingO_iR_i$.getAverage();
11:
12:     Array $O_o \leftarrow op$.getOutput().getSemantics();
13:     Array $ratingO_oR_o = \emptyset$      //max per $r_o$ ratings
14:     ...     //as in $ratingO_oR_o$
15:     double $ratingO_o \leftarrow ratingO_oR_o$.getAverage();
16:
17:     //apply weights for overal operation rating
18:     if $R_i \neq \emptyset \wedge R_o \neq \emptyset$
19:       $ratingOp \leftarrow w * ratingO_i + (1-w) * ratingO_o$;
20:     else if $R_i = \emptyset$
21:       $ratingOp \leftarrow ratingO_o$;
22:     else
23:       $ratingOp \leftarrow ratingO_i$;
24:     end if
25:     $OperationRatings$.add($ratingOp$);
26:   end for
27: end for
28: return $OperationRatings.sort(desc)$;
```

Algorithm 1. The Tomaco matching function performing DFS extraction of semantics, logic-based rating and applying weights

Algorithm 2 Logic Based

```
Input:
$r_x$: requested input;
$O_x$: Array of requested output;
$x$: input or output identifier;
Output: Array of $O_x$ ratings according to $r_x$
Constants:
$UpperRate$: high rating for desired matches e.g. 0.75;
$LowRate$: low rating for less desired matches e.g. 0.25;

1:  Array $ratingO_xr_x \leftarrow \emptyset$;
2:  for each offered $o_x \in O_x$
3:    if $o_x \equiv r_x$
4:      $rating \leftarrow 1$;    //Exact
5:    else if $o_x > r_x$    //superclass of $r_x$
6:      if $x$ = input
7:        $rating \leftarrow UpperRate$;    //Desired case
8:      else
9:        $rating \leftarrow LowerRate$;    //Less desired case
10:     end if
11:   else if $o_x < r_x$    //subclass of $r_x$
12:     if $x$ = output
13:       $rating \leftarrow UpperRate$;    //Desired case
14:     else
15:       $rating \leftarrow LowerRate$;    //Less desired case
16:     end if
17:   else
18:     $rating \leftarrow 0$;    //Fail
19:   end if
20:   $ratingO_xr_x$.add($rating$);
21: end for
22: return $ratingO_xr_x$;
```

Algorithm 2. The Tomaco logic-based rating strategy

requested outputs (line 13). Each vector value is the maximum rating score between the requested concept and all underlying offered concepts (lines 6-9). Consequently, the overall input and output ratings per offered operation are the average values of the corresponding vectors (lines 10, 15), to provide balance. Actually, other algorithms get the minimum instead of the average of vector values, but this is intuitively stricter and as seen during internal experiments with TC3, provides worse results. The final offered operation rating is the weighted sum of input and output ratings (lines 17-23).

Note here, that this different handling of input and output does not mean that their importance is different. Input versus output significance is a separate parameter given in the form of weight (line 18). Usually, this is set to 0.5 unless a user desires otherwise. Additionally, the proposed algorithm does not apply weights in case one of the two, input or output, is entirely not requested (lines 20-23). On the contrary, when input or output is requested but does not match (*Fail*), overall rating is reduced accordingly.

Finally, the algorithm does not further normalize per operation rating for service ratings. Instead, each operation is returned as a whole. This approach is based on the principle that service operations are self-contained methods of certain input and output, whereas services are containers of such methods. In other words, the algorithm essentially performs operation matching and is able to handle operations independently. A rating threshold can also be applied as filtering means to improve quality versus quantity of retrieved services. This threshold was not applied during evaluations, for completeness, but is rather suitable for the web application system.

Syntactic-on-Semantics Strategy

The Syntactic-on-Semantics (denoted as Syn-On-Sem) method's purpose is to compensate for mismatches of the logic-based method i.e. when classes are not related but their names are similar. Typically, the Syn-On-Sem strategy handles semantic annotations as plain textual expressions, applying text-similarity metrics. In other words, the strategy measures syntactic text-similarity between requested semantics and offered semantic annotations.

First of all, the algorithm itself obtains semantic annotations using the same DFS-traversing strategy as the previous method. However, this time, concept name extraction occurs, i.e. annotation string URIs are trimmed before '#' to get the local names. Consequently, R_i, R_o and O_i, O_o are matched to provide an operation rating for the requested input and output. The core text-similarity matching is provided by a library of textbook methods. Note that pseudocode is not provided for compactness, but rather explained here. The aforementioned changes occur just by substituting the logic-based method calls in Algorithm 1 by a suitable method for name extraction and measuring text-similarity.

After experimenting and analyzing different text-similarity methods, we selected the two methods most suitable in this context. Semantic names, like variable names in programming, follow different conventions to comply with the disallowance of spaces. For this reason, standards such as CamelCase, i.e. capitalization of the first letter of each word in a phrase, and snake_case, i.e. underscore between words, were long ago established in programming. These naming conventions are also met in SAWSDL-TC (54% of concepts). As a result, target words to be found in text are in fact substrings.

For such string instances, the algorithms Monge-Elkan [30] and Jaro [31] were found to be more suitable compared to others e.g. Cosine, Dice, Euclidean Distance, Jaccard and Levenshtein. Their high suitability is justified, as Monge-Elkan was especially designed to match atomic strings where words are delimited by special characters, while Jaro targets short strings such as names. In internal experiments, Monge-Elkan was found to rate desired, similar instances of such strings high (not lower than 1), as did Jaro (above 0.7). Hence, we have set the corresponding thresholds for a text-similarity match at 1 and 0.7 respectively.

Syntactic-on-Syntactics Strategy

The Syntactic-on-Syntactics (denoted as Syn-On-Syn) strategy completely disregards semantics on offered input and output, performing text-similarity on the names of target WSDL elements. Hence, it is the only fruitful strategy to perform when semantic annotations are few or absent (plain WSDL files). The algorithm is rather similar to the previous strategies except minor changes in the pseudocode of Algorithm 1. Instead of collecting semantics, the tree is again traversed collecting each element's name string. Those names are then compared to each r_i and r_o forming an overall rating and ranking of operations. The text-similarity algorithms employed here are again Monge-Elkan and Jaro, since syntactic description element names are again expressed in CamelCase, snake_case or similar conventions.

Hybrid Strategy

As all the aforementioned strategies examine different targets and use different heuristics, they can compensate for one another. The effectiveness of each method is highly dependent on the subjective definition of relevance per query according to the importance of various semantic relationships or other criteria. The proposed hybrid method was designed according to the requirements posed in the beginning of this section, and then calibrated further, based on experiments with SAWSDL-TC (Section 3.2).

According to the above, the logic-based method should get the highest priority, allowing syntactic methods to compensate for it. Most existing hybrid methods exhaust logic-based matching up to *Fail,* before exploring textual-similarity, or perform all methods and numerically combine them (e.g. in [11]). On the contrary, our hybrid strategy does not always perform and combine all sub-strategies, but rather examines syntactic measures only after a logic-based *Exact* match has not been found. In detail, the method assimilates the following rating function $f(o_x)$, for each examined offered element o_x:

$$f(o_x) = \begin{cases} 1, & o_x \equiv r_x \ \vee \ o_x \sim_t r_x \ \vee \ name(o_x) \sim_t r_x \\ 0.75, & (x = i \ \wedge \ o_x > r_x) \vee (x = o \ \wedge \ o_x < r_x) \\ 0.25, & (x = i \ \wedge \ o_x < r_x) \vee (x = o \ \wedge \ o_x > r_x) \\ 0, & otherwise \end{cases}$$

where text-similarity is denoted as $\sim_t$, the WSDL name of element x as $name(x)$, input as i and output as o.

The above cases are explored in a sequential manner. If an *Exact* match (either equal or equivalent class) fails, Syn-On-Sem ($o_x \sim_t r_x$) and then Syn-On-Syn ($name(o_x) \sim_t r_x$) are performed sequentially. Each text measure returns a binary value; a match is found if the per algorithm threshold is exceeded (1.0 for Monge-Elkan and 0.7 for Jaro, as previously described). Otherwise, if text-similarity is below threshold, the *Desired* and *LessDesired* logic checks occur, resulting in respective lower ratings (0.75 and 0.25 respectively).

Thus, a semantic *Exact* match or a syntactic match are considered equal and get a hybrid rating of 1. The principle behind this choice is that, intuitively, a syntactic similarity match means that the offered concept is exactly the one sought; not e.g. its parent, child or sibling. In internal experiments, this assumption was confirmed by setting the hybrid rating for a syntactic match in decimal values, between 0 and 1. The highest improvement in metrics was clearly achieved with the *Exact* rating of 1. Due to space restrictions, we only present the evaluation of the final proposed hybrid method in Section 3.2, where text-similarity, while poor on its own, manages to improve logic-based performance in the hybrid setting.

It can be speculated, that not exhausting logic-based matching before syntactic matching allows it to compensate for *Exact* false negatives. This claim is supported in the evaluation section where Tomaco performs higher than other heuristic hybrid methods. Alternatively, one could choose or learn which method to use in each case or how to numerically combine sub-strategies, as in adaptive methods (Section 2). However, in the scope of this work, we are following different principles, e.g. to not always combine sub-strategies.

3.1 Algorithm Implementation

The proposed algorithm has been implemented entirely in Java. The *easyWSDL* library [32] is used to retrieve modelReferences in a DFS manner, while modelReference attributes from *wsd;l:operation* and *wsdl:part* from *wsdl:type* are retrieved explicitly. The logic-based method relies on

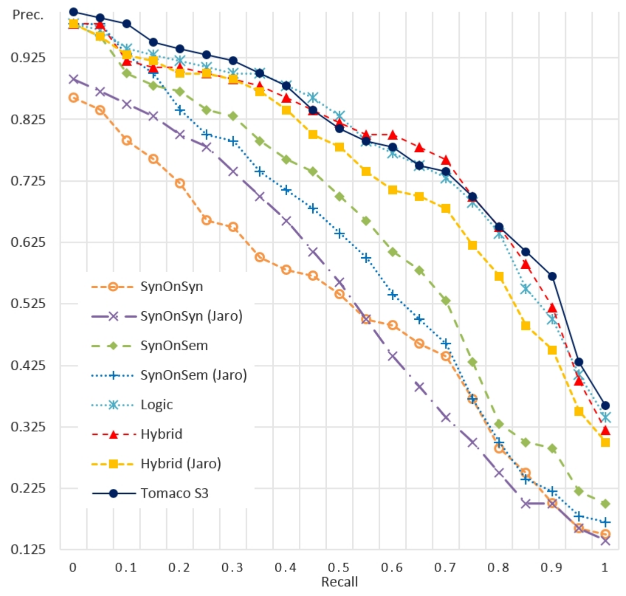


Fig 2. Macro-averaging precision of Tomaco variants

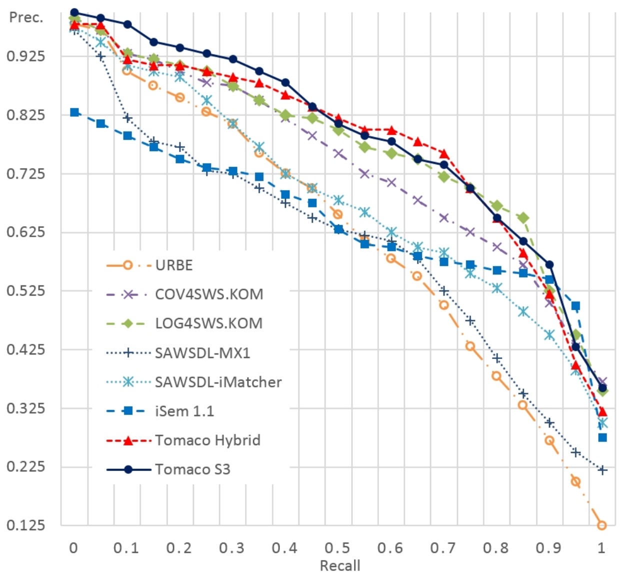


Fig 3. Macro-averaging precision of Tomaco Hybrid and Tomaco-S3 amongst S3 2012 participants

the OWL-API Java tool [33] to retrieve concepts and its underlying reasoner, Hermit [34], to evaluate their relationships. Text-similarity measures Jaro and Monge-Elkan are provided by the Simmetrics Java library[9].

Performance-wise, the implementation aims to optimize response time. Reasoner instances, known to introduce delays due to the task's complexity, are initialized at start-up and kept in memory, answering all queries on-demand. Traversing service descriptions introduced the longest delay. Therefore, extraction and indexing of all names and semantic terms in an internal structure (hash table) is also performed during initialization. The above optimizations resulted in having one of the top-performer systems, regarding response time, amongst state-of-the-art, as presented in the next subsection. Respective optimizations, more suitable to the integrated Tomaco web application, are presented in the following section.

3.2 Algorithm Evaluation

SAWSDL-TC3 [11] is one of the most eminent, uniform and large SAWSDL datasets publicly available for evaluation purposes. On top of that, it is integrated in a universal evaluation platform, SME2[10], used for the yearly S3 contest. Although the Tomaco algorithm was not designed with the particular dataset or contest in mind, SME2 offers the means to objectively evaluate it against state-of-the-art algorithms using universal metric implementations. Therefore, a Tomaco plugin[11] for SME2 was developed in order to conduct all experiments and to provide reproducibility. The plugin contains all proposed variants, plus the Tomaco-S3 variant, which specifically targets the contest. Tomaco-S3 syntactically compares query names with operation service names (rating them with the maximum), before performing the Tomaco hybrid method. While it improves rankings, it is naturally excluded from the open-ended Tomaco web application, where query names are not defined. The plugin was submitted to the S3 2014 call as a byproduct of this work.

An additional byproduct was a slightly, syntactically modified TC3[11], to allow loading the ontologies in popular tools used in Tomaco. Most notably, redundant imports in three (out of thirty-eight) ontologies were removed (importing one another multiple times caused problems in OWL-API). Note that not all other algorithms were available to re-evaluate with the modified set, nor is it known whether their internal tools are affected by such errors.

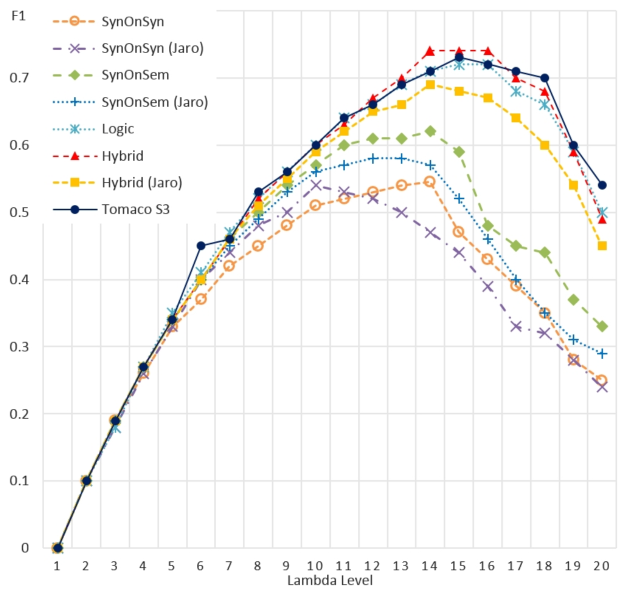


Fig 4. F-measure of Tomaco variants

[9] SimMetrics: http://sourceforge.net/projects/simmetrics/
[10] The SME2 tool: http://projects.semwebcentral.org/projects/sme2/
[11] Tomaco plugin, TC3 modification: http://lpis.csd.auth.gr/people/thanosgstavr/applications/tomaco.html

Table 3. Measures and performance of proposed Tomaco variants in SME2

	Logic-based	Syn-On-Syn	Syn-On-Syn (Jaro)	Syn-On-Sem	Syn-On-Sem (Jaro)	Hybrid	Hybrid (Jaro)	Tomaco-S3
AP	0.767	0.507	0.522	0.633	0.586	0.771	0.725	**0.785**
nDCG	0.851	0.594	0.67	0.749	0.742	0.838	0.826	**0.868**
Q	0.772	0.506	0.54	0.633	0.605	0.761	0.741	**0.795**
AQRT (s)	0.131	0.315	0.041	0.07	**0.029**	0.245	0.137	0.376
Total (m)	0.414	0.549	**0.344**	0.386	0.346	0.541	0.422	0.593

Table 4. Measures and performance of Tomaco and S3 2012 participants in SME2

	Tomaco-S3	Tomaco Hybrid	iSeM 1.1	LOG4SWS	COV4SWS	Nuwa	iMatcher	URBE	SAWSDL-MX1
AP	0.785	0.771	**0.842**	0.837	0.823	0.819	0.764	0.749	0.747
nDCG	0.868	0.838	0.803	**0.896**	0.884	0.884	0.855	0.85	0.839
Q	0.795	0.761	0.762	**0.851**	0.825	0.817	0.784	0.777	0.767
AQRT (s)	0.376	0.245	10.662	**0.241**	0.301	9.009	1.787	40.01	3.859

Given the minor correction, comparisons can be made with great confidence, until the next edition of the S3 contest.

To adapt Tomaco variants to the contest we set a rating threshold of zero, in order to return a ranking for all offered services. The input-vs-output weight was set to 0.5, since the organizers state that I/O carries the same significance. Textual similarity employs Monge-Elkan (threshold = 1) by default and Jaro (threshold = 0.7) when mentioned so. Overall, TC3 is compatible with the Tomaco process. It contains 1080 SAWSDL documents from nine domains: education, medical care, food, travel, communication, economy, weapons, geography and simulation. Each service has a single interface of a single operation, which suits the operation-centric ranking of Tomaco. *ModelReferences* to OWL2-DL ontologies are placed in *wsdl:parts* and underlying elements. The set includes 42 predefined queries for the contest in SAWSDL form and respective relevant sets in XML form. 38 OWL ontologies are used within services and queries alike.

Before presenting the evaluation on the entire set, two case studies are examined, in order to justify Tomaco's effectiveness and gain insight to the set's properties. The first case study compares syntactic properties of the set with respect to SynOnSyn and SynOnSem (Monge-Elkan). As opposed to the open-world scenario, the two methods are almost equivalent in a large portion of TC3, as most annotations have the exact syntactic names of elements. This is particularly apparent in query #26, where both methods have the exact precision of 0.842. However, they are both generally essential to compensate for one another. E.g. in queries #40, #11 syntactic names are meaningless, rendering SynOnSyn useless (precision 0.005, 0.003) while SynOnSem is still effective (precision 0.504, 0.756).

The second case study examines the performance gain of the chosen text-similarity algorithms. In particular, text-similarity in hybrid significantly increased precision over pure logic-based in cases such as #6 (precision 0.899 from 0.504) and #9 (precision 0.804 from 0.681). For example, in query #6, mid-level-ontology.owl#Prepared-Food is sought, while SUMO.owl#Food is relevant. While ontologies are not semantically linked, text-similarity retrieves the service, even at the absence of semantics.

The evaluation on the entire set initially examines all flavors of the proposed Tomaco variants. Macro-averaging precision at twenty recall levels (Fig 2) and F1 score on twenty lambda levels (Fig 4) both show the superiority of the S3 and Hybrid variants. The former exceeds overall, especially in early levels, while the latter exceeds in mid-levels. Table 3 shows that the same ranking holds for AP, while for nDCG and Q the logic-based variant ranks second, after S3 and before Hybrid. All measures dictate that, although pure textual methods perform much lower than logic-based, they do improve performance of the latter when combined in the Hybrid method. Additionally, the S3 variant improvement over Hybrid shows that query names indeed play a significant role in TC3. Internally, we also experimented with the impact of reasoning for semantically equivalent classes as *Exact* matches, which indeed improved logic and Hybrid methods (by 2% and 1% AP respectively). The experiments also show the significance of textual-similarity algorithm selection, as Jaro is found less accurate for the most part. We also internally experimented with other algorithms which significantly lowered AP (e.g. Levenshtein by 53% on Syn-On-Syn).

Performance-wise, the Jaro variants are faster than Monge-Elkan, while compromising precision. Meanwhile, the most accurate variants S3 and Hybrid manage to sustain a reasonably fast per query average response time (AQRT) and a total running time of around half a minute. Additional experiments were targeted to break total time down, outside the SME2 tool (which slightly alters running

Table 2. Time decomposition for Tomaco variants

	Logic-based	Syn-On-Syn	Syn-On-Sem	Hybrid
Total time	60.887s	**37.861s**	44.320s	71.370s
Init. Reasoners	11.638s	**0.000s**	0.000s	11.466s
DFS extraction	40.809s	**35.209s**	35.318s	39.499s
All queries	8.440s	**2.652s**	9.002s	20.405s
Per query avg.	0.201s	**0.063s**	0.214s	0.486s

time). Table 2 shows total time decomposition in reasoner initialization (if applicable), DFS-extraction, rating for all 42 queries and per query response time. Total query time for Syn-On-Sem is notably higher than for Syn-On-Syn, due to name extraction. All experiments were performed on an Intel i5 @3.20GHz, 8.00GB RAM.

Table 4 presents the two most effective variants, S3 and Hybrid, amongst state-of-the-art algorithms as presented on the S3 contest of 2012[5] [17], using the same tool and dataset. Tomaco variants rank after iSeM, LOG4SWS, COV4SWS and Nuwa [17] in AP, but above iMatcher, URBE and SAWSDL-MX1. In nDCG and Q-measure, only LOG4SWS, COV4SWS and Nuwa surpass Tomaco-S3. Meanwhile, Tomaco variants perform significantly better in macro-averaging precision at recall levels, shown on Fig 3. Tomaco-S3, especially, ranks above other algorithms for the most part, up to 0.7 recall level.

According to the nature of the metrics, Tomaco in general is not optimal, as per the metrics over total recall, e.g. MAP, on Table 4. However, it prevails in macro-averaging precision, since the metric measures each recall level separately and Tomaco outperforms others, especially at early ones (Fig 3). This is due to Tomaco successfully finding most, not all, results early i.e. at the top rankings. Hereby we conclude that the proposed algorithms are optimal when a portion, e.g. top-k, relevant services is required and sufficient (as a side-note, top-k AP is currently not available for comparison neither in SME2 nor in literature).

Additionally, the published per query response times rank Tomaco just after LOG4SWS, COV4SWS, hence, showing a fair trade-off of precision versus computational time. Please note that algorithms were not re-evaluated, neither do we know the specs of the S3 2012 environment, so performance is not directly comparable but is reported here only for the sake of completeness. Apparently, learning algorithms take significantly more time, with the exception of iMatcher and the top ranking ones. We also speculate that learning algorithms also require significant time for initialization i.e. training (please note that total times and technical specs of the S3 contest are not available at the time).

As a general remark, Tomaco does not rank first in overall metrics, but shows an optimal performance in macro-averaging precision. Apparently, it performs exceptionally well for a large percentage of recall levels, especially early ones, as it ranks most of the relevant documents high. Hence, it can be useful for use cases where the user demands most, but not all, relevant documents, while maintaining a satisfactory response time. While the evaluation only considers final, hybrid methods, some individual methods also show improvement e.g. Tomaco Logic over logic-based SAWSDL-MX0. Due to size and scope restrictions the reader is referred to literature for such comparisons [11].

4 Tomaco Web Application

The Tomaco web application is an integrative Tool for Matching and Composition of Web Services available on the Web[12], while this work focuses on the Matching part. The motivation behind this attempt is twofold: 1) to provide Web Service developers, researchers and consumers with a ready-to-use platform for matching new or existing services, in a user-friendly graphical manner 2) to promote the use of Semantic Web Services, SAWSDL/ WSDL and the Semantic Web altogether, targeting the lack of user-friendly, functional tools for these technologies (as acknowledged in [35]).

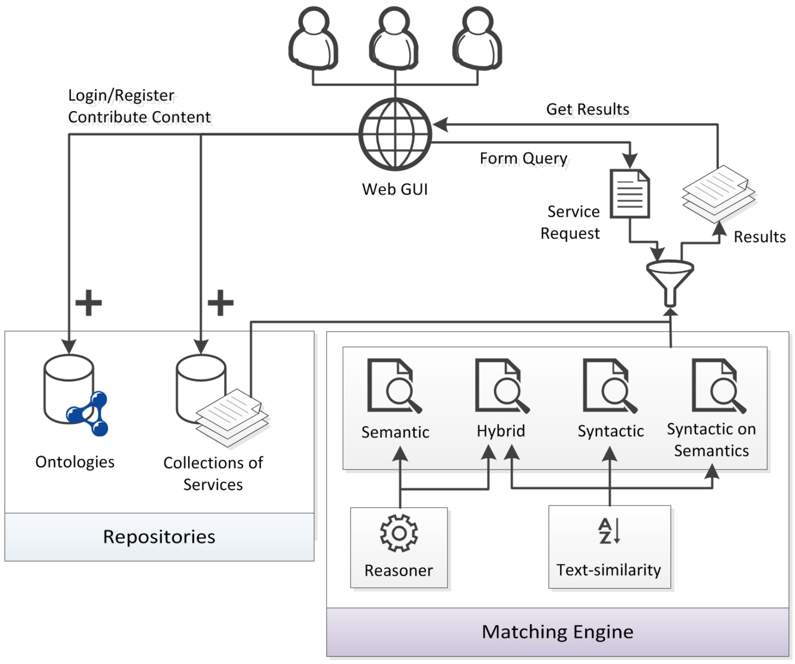


Fig 5. The Tomaco Web Application infrastructure, underlying repositories, matching engines and user interaction

[12] Tomaco web application: http://tomaco.csd.auth.gr

4.1 Tomaco Application Architecture

The system architecture, as shown in Fig 5, allows users to access all its functions through a web GUI. Regarding content, they can add files to the Ontology and Service repositories stored at the server. Regarding usage, they can form queries using ontological terms. Consequently, the matching subsystem provides strategies to match the query across Service storage and return a filtered ranking of results to the GUI. The embedded reasoner is used during logic-based and hybrid strategies within the matching engine. Likewise, a text-similarity algorithm library is used during Syntactic and Hybrid matching strategies.

4.2 Functionality

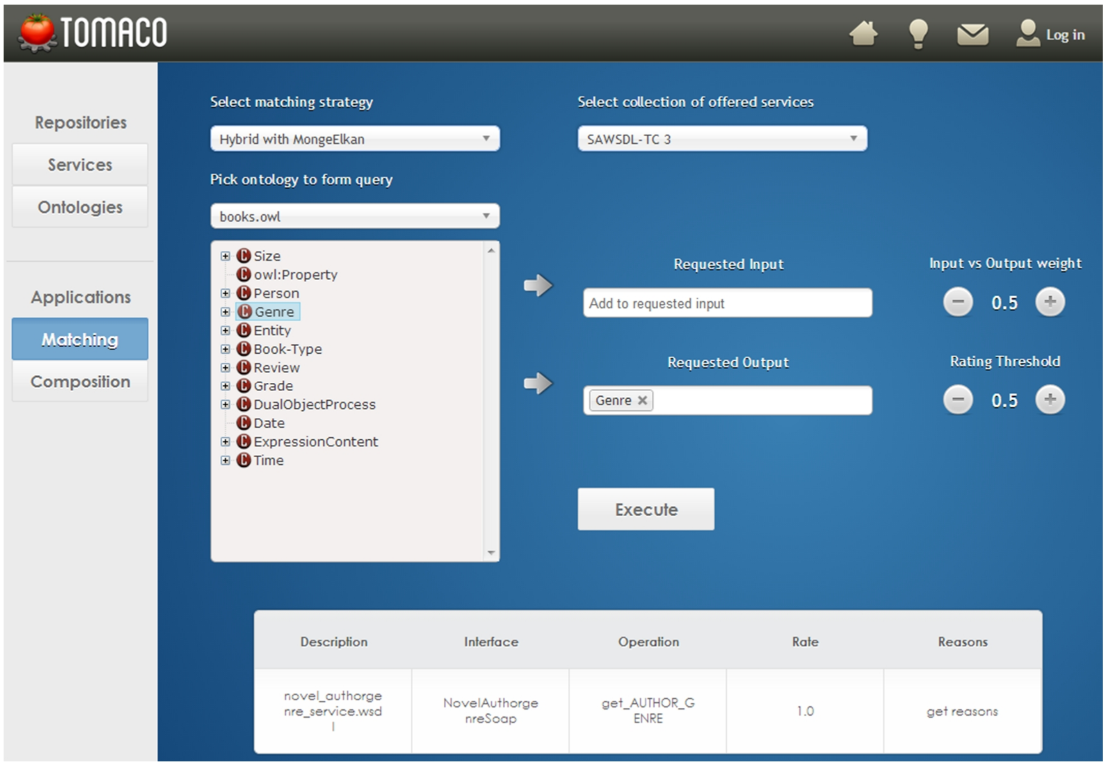
One major purpose of Tomaco is to provide a comprehensive Web Service registry, organized in collections. Collections mostly reflect the origin of their services, e.g. SAWSDL-TC3 constitutes a collection, although its services target different application domains. Users are able to browse existing service collections or register to create their own. Each collection holds metadata, such as uploader, date and description and can serve as a search frontier for matching or composition. New services can be uploaded from local user stores or linked from online sources. Service names can be browsed with autocomplete. Upon selecting a service, its definition is displayed in a tree hierarchy. Similarly to Services, Tomaco stores a collection of ontologies, necessary for forming queries. Ontologies can be browsed with Autocomplete, added using local files or online sources and used during matching or composition. Selecting an ontology displays its metadata and contents (classes and hierarchy).

The next segment of the application provides algorithms for automatic service matching and composition. The matching process integrates the algorithm presented in Section 3. A service collection is used as a search frontier i.e. offered services. To construct a query, i.e. provide R_i and/or R_o, users have to pick concepts by browsing ontologies in a graphical manner. Optional matching parameters include input-vs-output weight and rating threshold, both between 0.1 and 0.9.

Four strategies proposed in this work can be chosen as the matching method: Logic-based, Syn-On-Sem, Syn-On-Syn or Hybrid. Both Monge-Elkan and Jaro are applicable for syntactic and hybrid methods. Results are presented on a table, showing each operation's name, rating in descending order, service and interface. The user can finally view a justification for each rating, namely, the name, type and semantics of the underlying matched offered element with the highest rating. A comprehensive view of query formation and execution is depicted in Fig 6.

4.3 System Implementation

The Tomaco application is currently hosted on a single Apache Tomcat server. The server hosts service and ontology file copies locally for constant availability; even for those linked online. File metadata, such as descriptions, are stored in a MySQL database along with user profiles. WSDL files are parsed and indexed at upload time to boost retrieval. The matching subsystem employs all technologies and libraries as described in algorithm implementation, e.g. easyWSDL for parsing service descriptions, OWL-API for traversing ontologies, Hermit for reasoning



Select matching strategy: Hybrid with MongeElkan

Select collection of offered services: SAWSDL-TC 3

Pick ontology to form query: books.owl

- Size
- owl:Property
- Person
- Genre
- Entity
- Book-Type
- Review
- Grade
- DualObjectProcess
- Date
- ExpressionContent
- Time

Requested Input: Add to requested input

Input vs Output weight: 0.5

Requested Output: Genre

Rating Threshold: 0.5

Description	Interface	Operation	Rate	Reasons
novel_authorgenre_service.wsdl	NovelAuthorgenreSoap	get_AUTHOR_GENRE	1.0	get reasons

Fig 6. Web Service matching query formation and retrieval in Tomaco's graphical user interface

upon them and Simmetrics for text-similarity algorithms. The GUI was implemented using HTML, CSS, JavaScript and jQuery. JSP is used for e.g. retrieving local files, and JavaScript/jQuery for invoking servlets in the backend. Java servlets are used to retrieve database metadata, read and manage server files and to invoke the matching subsystem. In order to optimize real-time performance, Tomaco invokes DFS-extraction and indexing of service elements in the database (instead of in memory, as in the SME2 environment) on service-upload time. Reasoning is performed for all ontologies (as in the experiments) so as to maintain a low per query response time (in the order of a few seconds).

5 Conclusions and Future Work

This paper introduces a Semantic Web Service matching algorithm for SAWSDL, entailing three pure strategies, namely logic-based, Syntactic-On-Syntactics and Syntactic-On-Semantics, and a hybrid composite strategy. It also presents the integration of the algorithm in a web application named Tomaco, along with a service registry and an architecture to provide on-demand matching, performed on both existing and user-contributed content. The underlying algorithm's evaluation is carried out using the S3 contest environment, which allows reproducing results and positions Tomaco amongst state-of-the-art algorithms. The proposed logic-based method proves more effective than pure text-similarity strategies. However, text-similarity, with appropriate algorithms, does significantly improve and compensate for logic mismatches in the proposed hybrid variant. Meanwhile, Tomaco ranks high amongst state-of-the-art algorithms, especially for the initial recall levels i.e. when a portion of relevant services is required. Optimizations, such as indexing and preprocessing, resulted in a satisfactory low response time for the Tomaco web application.

Future work is mainly focused towards two directions: enriching the algorithm itself and extending the Tomaco system. The syntactic strategies can possibly benefit from developments in information retrieval, such as the Google or Flickr distance metrics. Regarding ontological similarity, we plan to examine concept distance and free-text descriptions (e.g. rdfs:label, rdfs:comment), which differ widely from names and require a study of proper algorithms. Depth, although acquired during DFS, requires a separate study to be considered, as to how it can be combined (e.g. to diminish numerical ratings) and in which cases. From another perspective, since queries and relevancy are subjective, voting techniques could ensure finding the optimum strategy in each case.

On enriching the Tomaco web application, we plan to focus on defining motivation and innovative methods towards user-accessible service composition. Technically, its overall usability can be improved by providing a REST API for invoking the system's functions and allow software agents to discover the services, while exploring further integration with existing service registries and providers. Finally, performing a preliminary user acceptance study, collecting feedback and community interaction are critical towards improving the platform.

6 Acknowledgments

The authors wish to thank Mr. Thodoris Tsompanidis for his earlier work, Dr. Efstratios Kontopoulos, Dr. George Meditskos and the reviewers for their fruitful comments.

7 References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.

[2] K. Iqbal, M. L. Sbodio, V. Peristeras, and G. Giuliani, "Semantic service discovery using SAWSDL and SPARQL," in *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*, 2008, pp. 205–212.

[3] F. Lécué, S. Salibi, P. Bron, and A. Moreau, "Semantic and syntactic data flow in web service composition," in *Web Services, 2008. ICWS'08. IEEE International Conference on*, 2008, pp. 211–218.

[4] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, and M. Solanki, "Bringing semantics to web services: The OWL-S approach," in *Semantic Web Services and Web Process Composition*, Springer, 2005, pp. 26–42.

[5] H. Lausen, A. Polleres, and D. Roman, "Web service modeling ontology (wsmo)," *W3C Memb. Submiss.*, vol. 3, 2005.

[6] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "Sawsdl: Semantic annotations for wsdl and xml schema," *Internet Comput. IEEE*, vol. 11, no. 6, pp. 60–67, 2007.

[7] T. Vitvar, J. Kopeckỳ, J. Viskova, and D. Fensel, "Wsmo-lite annotations for web services," in *The semantic web: Research and applications*, Springer, 2008, pp. 674–689.

[8] F.-Z. Belouadha, H. Omrana, and O. Roudiès, "A model-driven approach for composing SAWSDL semantic Web services," *ArXiv Prepr. ArXiv10043256*, 2010.

[9] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, and S. Pettifer, "BioCatalogue: a universal catalogue of web services for the life sciences," *Nucleic Acids Res.*, vol. 38, no. suppl 2, pp. W689–W694, 2010.

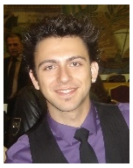
[10] I. Katakis, G. Meditskos, G. Tsoumakas, N. Bassiliades, and Vlahavas, "On the Combination of Textual and Semantic Descriptions for Automated Semantic Web Service Classification," in *Artificial Intelligence Applications and Innovations III*, Iliadis, Maglogiann, Tsoumakasis, Vlahavas, and Bramer, Eds. Springer US, 2009, pp. 95–104.

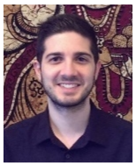
[11] M. Klusch and P. Kapahnke, "Semantic web service selection with SAWSDL-MX," in *The 7th International Semantic Web Conference*, 2008, p. 3.

[12] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 7, no. 2, pp. 121–133, 2009.

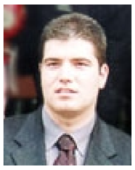
[13] F. Kaufer and M. Klusch, "Wsmo-mx: A logic programming based hybrid service matchmaker," in *Web Services, 2006. ECOWS'06. 4th European Conference on*, 2006, pp. 161–170.

[14] M. Klusch, P. Kapahnke, and I. Zinnikus, "Sawsdl-mx2: A machine-learning approach for integrating semantic web service matchmaking variants," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 335–342.
[15] M. Klusch and P. Kapahnke, "isem: Approximated reasoning for adaptive hybrid selection of semantic services," in *The semantic web: Research and applications*, Springer, 2010, pp. 30–44.
[16] U. Lampe and S. Schulte, "Self-Adaptive Semantic Matchmaking Using COV4SWS. KOM and LOG4SWS. KOM," in *Semantic Web Services*, Springer, 2012, pp. 141–157.
[17] M. Klusch, "Overview of the S3 contest: Performance evaluation of semantic service matchmakers," in *Semantic Web Services*, Springer, 2012, pp. 17–34.
[18] D. Wei, T. Wang, J. Wang, and A. Bernstein, "SAWSDL-iMatcher: A customizable and effective Semantic Web Service matchmaker," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 9, no. 4, pp. 402–417, Dec. 2011.
[19] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis, "Serving the Sky: Discovering and Selecting Semantic Web Services through Dynamic Skyline Queries," in *2008 IEEE International Conference on Semantic Computing*, 2008, pp. 222–229.
[20] V. X. Tran, S. Puntheeranurak, and H. Tsuji, "A new service matching definition and algorithm with SAWSDL," in *3rd IEEE International Conference on Digital Ecosystems and Technologies, 2009. DEST '09*, 2009, pp. 371–376.
[21] C. Fellbaum, *WordNet*. Springer, 2010.
[22] G. Meditskos and N. Bassiliades, "Object-Oriented Similarity Measures for Semantic Web Service Matchmaking," in *Fifth European Conference on Web Services, 2007. ECOWS '07*, 2007, pp. 57–66.
[23] G. Meditskos and N. Bassiliades, "Structural and role-oriented web service discovery with taxonomies in OWL-S," *Knowl. Data Eng. IEEE Trans. On*, vol. 22, no. 2, pp. 278–290, 2010.
[24] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci, "IRS-III: A broker-based approach to semantic Web services," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 6, no. 2, pp. 109–132, 2008.
[25] J. Becker, O. Mueller, and M. Woditsch, "An Ontology-Based Natural Language Service Discovery Engine–Design and Experimental Evaluation," 2010.
[26] P. Plebani and B. Pernici, "URBE: Web service retrieval based on similarity evaluation," *Knowl. Data Eng. IEEE Trans. On*, vol. 21, no. 11, pp. 1629–1642, 2009.
[27] U. Kuster, B. Konig-Ries, and A. Krug, "Opossum-an online portal to collect and share sws descriptions," in *Semantic Computing, 2008 IEEE International Conference on*, 2008, pp. 480–481.
[28] O. Hatzi, G. Batistatos, M. Nikolaidou, and D. Anagnostopoulos, "A Specialized Search Engine for Web Service Discovery," in *2012 IEEE 19th International Conference on Web Services (ICWS)*, 2012, pp. 448–455.
[29] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue, "iServe: a linked services publishing platform," in *CEUR workshop proceedings*, 2010, vol. 596.
[30] A. E. Monge and C. Elkan, "The Field Matching Problem: Algorithms and Applications.," in *KDD*, 1996, pp. 267–270.
[31] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *J. Am. Stat. Assoc.*, vol. 84, no. 406, pp. 414–420, 1989.
[32] N. Boissel-Dallier, J.-P. Lorré, and F. Benaben, "Management Tool for Semantic Annotations in WSDL," in *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, R. Meersman, P. Herrero, and T. Dillon, Eds. Springer Berlin Heidelberg, 2009, pp. 898–906.
[33] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.
[34] R. Shearer, B. Motik, and I. Horrocks, "HermiT: A Highly-Efficient OWL Reasoner.," in *OWLED*, 2008, vol. 432.
[35] E. Della Valle, G. Niro, and C. Mancas, *Results of a Survey on Improving the Art of Semantic Web Application Development*. SWESE, 2011.


Thanos G. Stavropoulos is a PhD Student at the Dept. of Informatics at the Aristotle University of Thessaloniki (AUTH), after receiving his BSc (2009) and MSc (2011) degrees from the same department. Meanwhile, he has worked in projects such as Dem@Care FP7 (at the Centre for Research and Technology Hellas) and Smart IHU (at the International Hellenic University). His research interests include intelligent autonomous systems, semantic web services, ambient intelligence and sensor networks, while he is Chair of the ACM Student Chapter of AUTH. [http://users.auth.gr/athstavr]


Stelios Andreadis has received his BSc (2011) and MSc (2014) degrees in Information Systems, both from the Dept. of Informatics of the Aristotle University of Thessaloniki (AUTH), receiving scholarships for excellence. His research interests include Semantic Web Services, Information Retrieval and Machine Learning, while he is also Treasurer of the ACM Student Chapter of AUTH.

Nick Bassiliades is an Associate Professor at the Dept. of Informatics at the Aristotle University of Thessaloniki, Greece, after receiving his PhD degree in parallel knowledge base systems (1998). His research interests include knowledge-based and rule systems, agents, ontologies, linked open data and the Semantic Web. He has published more than 150 papers in journals, conferences, and books, and coauthored two books, receiving over 1200 citations (h-index 19). He has been involved in 30 R & D projects, 90 conferences (as PC) and co-chaired RuleML symposia and WIMS-2014. He is a director of RuleML, Inc. and member of the Greek Computer Society, IEEE, and ACM.[http://tinyurl.com/nbassili]


Dimitris Vrakas is a Lecturer at the Dept. of Informatics at the Aristotle University of Thessaloniki, Greece, after receiving his PhD degree in Intelligent Planning Systems from the same department in 2004. His research interests include artificial intelligence, intelligent tutoring systems, planning, heuristic search and problem solving. He has published more than 50 papers in journals and conferences, and co-edited 2 books. He has been involved in projects concerning intelligent agents, e-learning and web services [http://lpis.csd.auth.gr/vrakas].

Ioannis Vlahavas is a Professor at the Dept. of Informatics at the Aristotle University of Thessaloniki, after receiving his Ph.D. degree in Logic Programming Systems from the same department (1988). He specializes in machine learning, knowledge-based and AI systems and has published over 250 papers and 9 books, receiving 3700 citations (h-index 30). He has been involved in more than 30 projects and chaired many conferences [http://www.csd.auth.gr/~vlahavas].