Subsumption Reasoning for QoS-Based Service Matchmaking

Kyriakos Kritikos, Dimitris Plexousakis

Subsumption Reasoning for QoS-based Service Matchmaking

Kyriakos Kritikos and Dimitris Plexousakis

ICS-FORTH
Heraklion GR-70013, Greece
{kritikos, dp}@ics.forth.gr

Abstract. Service-orientation has revolutionized the way applications are constructed and provisioned. To this end, a proliferation of web services is being increasingly available. To exploit such services, an accurate service discovery process is required with a suitable performance focusing both on functional and quality of service (QoS) aspects. In fact, QoS is the main distinguishing factor for the plethora of functionally-equivalent services available in the internet. Accuracy in service discovery comes via exploiting formal techniques and ontologies in particular. Satisfactory performance levels can be reached via using smart methods that intelligently organise the service advertisement space. In this paper, we propose smart ontology-based QoS-aware service discovery algorithms that exploit ontology subsumption as a means of matching QoS requests and offers. These algorithms exploit a variety of methods to structure the service advertisement space. Based on the empirical evaluation conducted, our proposed algorithms outperform the state-of-the-art in certain circumstances. To this end, ontology-based subsumption is indeed a promising technique to realise QoS-based service matchmaking.

Keywords: service, matchmaking, discovery, QoS, ontology, subsumption

1 Introduction

Service-orientation has revolutionalized the way web applications and processes are constructed, provisioned and evolved. With the advent of cloud computing, which delivers extra advantages, a proliferation of available services has been achieved covering various types of functional capabilities. To exploit such services and rapidly build added-value functionality, there is a need for accurate and fast service discovery algorithms focusing both on functional and quality-of-service (QoS) aspects. The state-of-the-art in functional service discovery exploits either ontology-based [9], information retrieval [2] or a mixture of such techniques [7] to perform service matching. It has been proven that only when ontology-based techniques are involved [7], higher accuracy levels can be attained.

However, functional service discovery alone cannot enable the service designer to discover those services satisfying all requirement aspects. On the contrary, QoS has been deemed as the aspect enabling the differentiation between

the plethora of functionally-equivalent services currently available. In fact, QoS can play a significant role in all phases of the service lifecycle [3]. To this end, various types of QoS-based service discovery approaches have been proposed. To increase the accuracy in the service discovery results, some of these types do exploit either ontology-based techniques [11] alone or constraint solving techniques as well [5]. Those approaches exploiting solely ontology-based techniques use ontology-based subsumption to perform the matching but have relied on wrong ontology constructs to specify QoS-based service specifications. As such, their applicability is quite limited. On the other hand, mixed-based approaches have a wider applicability and have been shown to exhibit much better performance.

In this paper, we propose a pure ontology-based approach which exploits ontologies in a correct way via more suitable constructs enhancing the respective applicability. In addition, we propose smart algorithms which intelligently organise the service offer space so as to perform service matchmaking via ontology subsumption on a subset of all offers. By considering the two main disadvantages of a mixed-based approach which are the ontology to constraint specification transformation and the solving of multiple constraint models to infer the matching between a pair of a service QoS offer and demand, our empirical evaluation shows that our proposed algorithms outperform mixed-based state-of-the-art ones in certain circumstances. This is a proof that ontology subsumption alone can be considered as a promising technique for QoS-based service matchmaking.

The rest of the paper is structured as follows. Section 2 reviews the related work. Section 3 provides background knowledge enabling to better understand the paper propositions. Section 4 analyses the proposed approach and the algorithms realising it. Section 5 discusses the experimental evaluation results. Finally, Section 6 concludes the paper and draws futher research directions.

2 Related work

2.1 QoS-based service description

A plethora of languages have been proposed for describing QoS-based offers and requests. According to the survey in [3], these languages can be distinguished according to their formality, expressiveness and complexity. From these languages, OWL-Q [4], a modular and semantic-based service description language, seems to be the most promising, especially in terms of expressiveness as it covers in a rich manner all possible aspects of QoS-based service description. Based on OWL-Q, a mid-level ontology is available which provides a common vocabulary of QoS terms which can be used to populate QoS-based service specifications, such as domain-independent QoS attributes and metrics (e.g., *response time*). Due to the above unique advantages, the approach proposed in this paper exploits OWL-Q along with its mid-level ontology.

2.2 QoS-based service discovery

Various QoS-based service matchmaking approaches have been proposed that can be categorised in three main types. Ontology-based approaches [11] rely on ontology subsumption to perform the service specification matching. These approaches are able to support only unary-based service specifications, i.e., involving one QoS term per constraint (QoS capability or requirement). Constraint-based approaches [1] assume the existence of a common QoS term vocabulary through which constraint models can be specified mapping to the actual service QoS offers and requests. Then, they exploit constraint solving techniques and specific matchmaking metrics to perform the matching of the constraint models. In comparison to ontology-based approaches, constraint-based ones operate over n-ary specifications and have a much better performance. Finally, mixed based approaches [5] attempt to exploit the best of both worlds. This means that they operate over semantic QoS specifications by first aligning them according to their QoS terms and then transforming them into constraint models which can then be matched based on the second type of approaches. In comparison to the former two types, this type can operate on n-ary specifications, it exhibits better accuracy levels due to the alignment of the specifications that goes beyond using subsumption reasoning and exhibits almost equivalent performance levels with respect to the constraint-based approach type.

Apart from the above approach categorisation, recently some new mixed approaches [6] have been proposed able to speed up the service matching time by cleverly organising the QoS service offer space. These approaches create a QoS offers subsumption hierarchy. As such, when a QoS request subsumes a hierarchy node, it subsumes all its descendants. Thus, these descendants do not have to be matched with the QoS request and matching time gets reduced.

The approach proposed in this paper belongs to the first type. As such, it suffers from the disadvantage of handling only unary constraints. However, this disadvantage is not crucial as most, if not all, of existing QoS service specifications in the real world are unary. Moreover, to speed up the matchmaking time, we propose different algorithms which attempt to similarly organise the offer space as in the recent approaches. In this way, we do not only reduce the matchmaking time but are able to outperform these recent approaches in certain cases for the following two reasons: (a) our approach does not require transforming ontology-based QoS specifications to a different form and (b) ontology subsumption can be faster than constraint-based matching even for a pair of a QoS offer and demand due to the way constraint matching metrics are realised. Thus, the use of pure ontology-based approaches in QoS-based service matching is not only feasible but also quite practical in certain cases. We believe that the prospective practitioners will benefit from the proposal and findings of this paper.

3 Background

In this section, we first explain why the current realisation of the pure ontology-based approach type is not appropriate and what is our proposal for solving

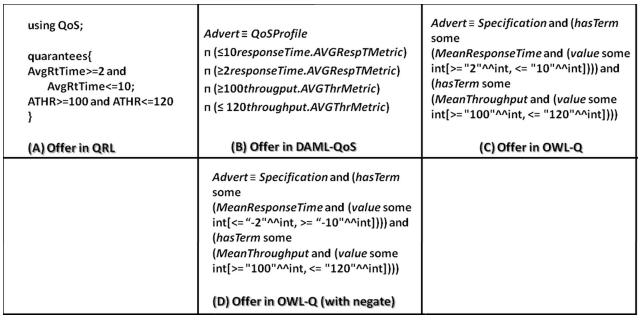


Fig. 1: The example QoS offer in different forms

this issue. Then, we highlight what is the process for adding and matching QoS offers for both ontology-based approach types as this paper focuses on their comparison.

3.1 Realisation Issues

Suppose we have the example QoS offer in Figure 1.A in QRL [1]-like syntax indicating that *average response time* will be less or equal to 10 and greater than 2 seconds while *average throughput* will be between 100 and 120 requests per second inclusive. In the pure ontology-based approach type, the two metrics will originate from a mid-level ontology specifying domain-independent QoS terms. Indeed, this is the case of the approach in [11] which maps both metrics to subclasses of *CompositeMetric*, thus connecting the upper-level ontology proposed to the mid-level one. However, the latter approach will then rely on a misuse of OWL cardinality constraints to specify the constraints of the specifications as indicated in Figure 1.B. In particular, it will indicate that the cardinality of the value-based properties (e.g., *responseTime*) of these metrics will be in accordance to the ranges in Figure 1.A. This wrong QoS constraint modelling has two main disadvantages: (a) only a specific type of QoS terms can be addressed mapping to non-negative integers – as such, terms like availability cannot be catered as their value types map to real numbers; (b) this modelling can also lead to an error that is at maximum one-half of the QoS term unit. Apart from the wrong modelling, the respective QoS ontology language exploited is quite limited with respect to the capabilities and richness of OWL-Q.

The above modelling issue is solved via the rationale in Figure 1.C. While there are again terms mapping to a composite metric class, we actually restrict the range of the *value* datatype property for them based on the desired limits. As such, we can model different value types for these terms; either concrete XSD[1] types (e.g., *integer*) or specialisations of them (e.g., constrained integers). By also exploiting OWL-Q, our QoS modelling is correct, richer and more extensive.

[1] `http://www.w3.org/TR/xmlschema11-1/`

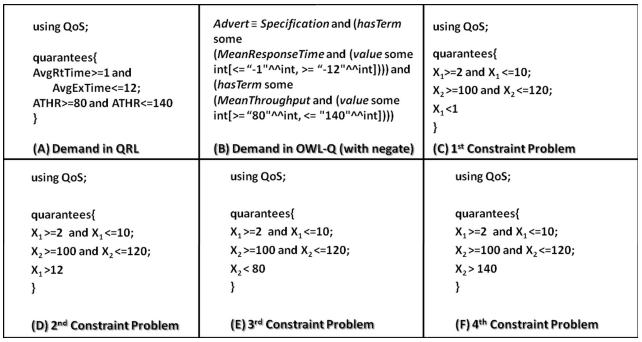


Fig. 2: The QoS demand and the 4 matchmaking problems

However, there is still a specific issue. Subsumption reasoning caters mainly for positively monotonic QoS terms. This can be understood from the example QoS demand in Figure 2.B which needs to be matched with the aforementioned QoS offer. While it is apparent that the QoS offer is more specific than the QoS demand and there is a match, the *average response time* is a negatively monotonic metric. As such, any ontology reasoner will never infer that the QoS offer is subsumed by the QoS demand.

The solution to this problem is to negate the constraints on negatively monotonic metrics. This is equivalent to considering a new, positively monotonic term equal to the negation of the original term. In this way, both the QoS offer and demand will be expressed as in Figure 1.D and 2.B and their matching will be derived through an ontology reasoner, like Pellet[2] [8]. This treatment of QoS specifications has the following alternative concequences: (a) either the modeller should specify the QoS constraints as well as the QoS terms in the newly prescribed way taking special case on negatively monotonic terms or (b) the respective tools enabling the editing of the QoS specifications should be realised to transform internally the modeller constraints in the appropriate format or pre-processing of QoS specification via transformation tools is performed before the actual registration or matching of the specification is performed.

3.2 Ontology-based QoS Specification Management Process

In a pure ontology-based approach, the QoS offer management process is quite simple. The existence of a semantic repository is assumed where the support for a specific ontology language like OWL is offered. Obviously, on top of OWL, a QoS-based service description language like OWL-Q lies via which the offers are actually specified. As indicated in [6], a mixed-based approach requires a QoS term (needed for their alignment) and a constraint model repository. As such, extra storage requirements are imposed.

[2] `https://github.com/Complexible/pellet/`

QoS Offer Registration. In an ontology-based approach, QoS offers to be registered are first loaded in order to check whether they are consistent. This maps to creating a small knowledge base (KB) out of this offer and checking with this KB is consistent by evaluating whether any concept is subsumed by OWL *Nothing*. In a mixed-based approach, apart from consistency checking, the ontology needs first to be realised and validated and then to be transformed into a constraint model. Thus, it is expected that the offer registration time is faster in an ontology-based approach.

By considering the previous sub-section's example, the mixed-based approach will map the QoS offer in Figure 1.B into a constraint model similar to that of Figure 1.A where each unique QoS term will be mapped to a specific variable.

QoS Request Matching. A QoS request passes the same sub-process when issued as it must be checked for consistency. Then, it must be matched with all the QoS offers stored in the respective repository. In the pure ontology-based approach, the QoS request is entered into the existing KB and then classification is performed such that all subsumption relations are discovered between this request and all QoS specifications. As such, the QoS request is matched with all QoS offers that it subsumes. Please note that matchmaking as conformance or subsumption is the main matching metric in all approach types.

For the rest of the approach types, there is a matching of two constraint models mapping to the QoS request and offer when the solution space of the latter is included in the solution space of the former. This is translated in solving one or more constraint problems depending on the constraint arity of the QoS specifications. In case of unary constraints, specification conformance maps to checking M (mapping to the offer's number of constraints) constraint problems constructed by the offer's constraint model and a negation of each demand's constraint. If all problems are infeasible, then there is a match between the QoS offer and request. In case of n-ary constraints, only one constraint problem needs to be solved constructed from the QoS offer and the negation of the QoS demand. A match is inferred if the latter problem is infeasible. In the first case, more steps must be performed with respect to the pure ontology-based approach whose timing depends on the number of constraints involved. In the second case, one complicated step is performed but has no counterpart in the pure ontology-based approach as the latter can address unary constraints only.

In a mixed-based approach, the QoS request of the previous subsection will be first mapped to the constraint model in Figure 2.D. Then, four constraint problems need to be solved as depicted in Figures 2.C and 2.F.

4 Proposed Approach

4.1 Architecture

Figure 3 depicts the proposed approach architecture by visualising both the respective components and their interactions. There are five main components involved. The *Semantic Repository* is an ontology-based repository able to store

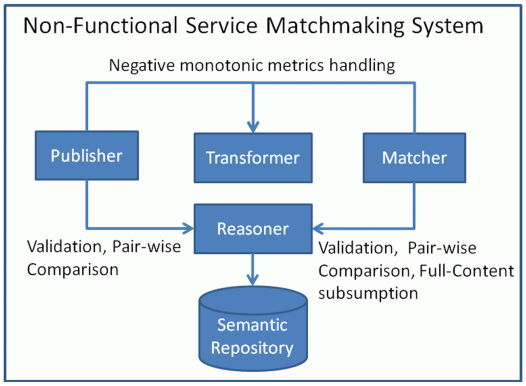


Fig. 3: Non-Functional Service Matchmaking System Architecture

all QoS offers. The *Matcher* is a web service (WS) taking as input a QoS request which is then matched with all QoS offers stored. This WS internally realises one or more matching algorithms so it can be configured to operate based on one of them. The *Publisher* is also a WS enabling service providers to publish their QoS offers in the semantic QoS-based matchmaking system. Similarly to *Matcher*, it can be configured to operate a specific (de-)publication algorithm mapping to the approach followed for matchmaking. Both the *Publisher* and *Matcher* exploit two other components: (a) the *Transformer* which loads the specification and then transforms the constraints for negatively-monotonic terms, when users do not model them as expected; (b) the *Reasoner* which performs different types of tasks: (a) ontology-based specification validation, (b) pair-wise specification comparison and (c) (incremental) subsumption over full repository content. Invalid specifications are returned back to their issuers with a suitable error message.

The semantic QoS-based matchmaking system currently operates solely on OWL-Q based specifications. In the future, it will include extra transformation functionality to support original specifications in different QoS-based service specification languages which will be injected in the *Transformer*'s existing capabilities. Distribution of content will also be examined to cater for better scalability levels. Due to the nature of the proposed algorithms, such distribution is quite easy and natural to realise without any implication on algorithm accuracy.

4.2 Algorithms

In the following, we are going to analyse the four main algorithms that we have realised and are included in the capabilities of the *Matcher* component. We focus on the two main processes supported: *QoS offer registration* and *QoS request matching*. For each algorithm, the presentation starts with the main rationale, it then explicates the way the offer space is organised, next the algorithm core is analyzed and finally its complexity analysis is supplied.

Naive *Rationale.* The main rationale of the algorithm, also justifying its name, is to load all offers on memory when a specific matching request is issued. This facilitates an offer's registration as once its consistency is checked, it is just stored in the repository. However, it is expected that matching time will not be appropriate as it has been proven that subsumption reasoning does not scale well, especially if done in a centralised manner. In fact, as OWL-Q along with its mid-level lies in the *SROIQ(D)* family of ontologies, any reasoning task is decidable but NExpTime-Hard.

Offer Space Organisation. There is no special offer space organisation. Only one specific hash set is employed to account for the URIs of the offers stored such that they can be immediately located and loaded during request matching.

Algorithm Core. The algorithm's core does not differ with respect to that sketched in Section 3. Initially, a KB is constructed out of all QoS offers and the QoS request and then classification is performed. Finally, a query on the KB is performed to obtain all offers subsumed by the request.

Complexity Analysis. Suppose that a specification usually has 4-5 QoS terms and 4-5 constraints on them, all wrapped into a single class definition. As such, we expect that the specification loading time will be more or less constant and equal to L_{spec}. Thus, the time needed to check the request's consistency (similar as loading) as well as construct a KB out of N offers and 1 QoS request will be $O(L_{spec} * (N + 2))$. If we further assume that the time to classify $N + 1$ specifications is S_{N+1}, such that the classification depends on the specification number, the overall matching time would be $O(L_{spec} * (N + 1)) + S_{N+1}$. We expect that usually $O(S_{N+1})$ takes much longer than $O(L_{spec} * (N + 2))$, especially when the specification number becomes bigger, so we will have a final complexity of $O(S_{N+1})$ for matchmaking. Offer registration, on the other hand, takes $O(N * L_{spec})$ time as each offer is just loaded and checked for consistency.

Incremental *Rationale.* The naive approach does not incrementally build the KB but constructs it on demand. As such, as incremental classifiers are available, it might be better to employ incremental classification to save time when classifying a temporal extension of the KB encompassing the request.

Offer Space Organisation. The previous algorithm's hash set is preserved to account for the offers already stored. KB is the other organisation medium constantly updated. The classification tree constructed contains all possible subsumption connections between the specifications involved.

Algorithm Core. During offer registration we discovered that it is a little bit costly to run classification each time an offer must be registered. As such, we run classification only every X offers, where X is a configuration parameter for the algorithm. In each registration, the offer is loaded, checked for consistency and then stored in the KB.

For request matching, after consistency checking, we temporarily include the request in the KB and then we perform incremental classification. We then query the KB to find the offers subsumed by the request.

Complexity Analysis. Suppose that S_X^Y is the incremental classification time when X specifications are added to the KB and Y specifications are already loaded. Then, the offer registration time for N offers will be $O\left(L_{spec} * N + \sum_{Y=1}^{\frac{N}{X}} S_X^{Y*X}\right)$.

On the other hand, the request matching time will take at most $O\left(L_{spec} + S_1^N\right)$ as we will have to check the request consistency and incrementally classify only the temporal addition of the request in the existing KB.

Subsumes *Rationale.* As a naive approach does not scale well in practice and driven by the fact that even the incremental algorithm might also exhibit similar performance problems, we decided to rely on the method in the *SubMIPMM* algorithm [6] and create our own subsumes offer hierarchy to be matched against any issuing request. In such hierarchy, if the request subsumes a node, then it also subsumes its descendants so some comparisons are avoided.

Offer Space Organisation. We do not construct a complete subsumes hierarchy as this requires connecting a new offer to all parents that subsume it and the registration time would be highly increased. The main trick as followed in [6] is to connect the offer to the first tree in the hierarchy forest in which it is matched.

Algorithm Core. The registration process is simple. We first match the new offer with all hierarchy's top offers. In case of a match, we check subsumption direction. If the new offer subsumes one or more top nodes, it becomes a top node itself and the matched nodes its children. If the offer is subsumed by one or more top nodes, we take the first one and check where to place the new offer in its own tree. So, the same matching procedure is followed until either the new offer subsumes some nodes in the selected tree or becomes this tree's leaf.

Concerning request matching, we match the request with all top-nodes in the subsumption hierarchy. In case the request subsumes a top-most node, we add this node along with its descendants in the matching offers set. Otherwise, we need to go down a top-most node's subtree similarly to the way top-matching is performed to find matching offers. The latter is due to the fact even if the request does not match the top-node, as we descend the tree, the offers becomes stricter with a smaller solution space and thus the probability that they finally match the request becomes higher.

For both processes, if pair-wise subsumption reasoning takes less than pair-wise constraint-based matching, this algorithm will be faster than *SubMIPMM*.

Complexity Analysis. Concerning offer registration, we need first to check offer consistency. Then, different cases can occur. In the best case, the offer is equivalent to the first top-most node so we do not need to check anything else. The time complexity will then become: $O\left(L_{spec} + S_2\right)$. In the worst case, the hierarchy maps to a tree and we have to put the new offer as a child of the rightmost leaf node. This means that we will have to compare the new offer with all offers stored. In this case, the time complexity is $O\left(L_{spec} + N * S_2\right)$ which can be reduced to $O\left(N * S_2\right)$. In the average case, B trees more or less balanced will exist and the time complexity will become: $O\left(L_{spec} + \frac{N+B^2}{2*B} * S_2\right)$ which can be reduced to $O\left(\frac{N+B^2}{2*B} * S_2\right)$.

Different cases map to request matching. The best one occurs when the hierarchy maps to a tree and the request subsumes the root node. The time complexity will be: $O\left(L_{spec} + S_2\right)$. The worst case occurs when the request must be compared with all tree nodes (as it does not subsume any offer or just the rightmost leaf one). The time complexity will be: $O\left(L_{spec} + N * S_2\right)$ which can be reduced to $O\left(N * S_2\right)$. In the average case, we assume that P offers will be subsumed and that there will be at least a two-level hierarchy between the subsumed offers. As such, the time complexity will be $O\left(L_{spec} + N * \left(1 - \frac{P}{2}\right) * S_2)\right)$ which is reduced to $O\left(N * \left(1 - \frac{P}{2}\right) * S_2)\right)$.

SubsumesFrag *Rationale.* The previous algorithm constructed the hierarchy in an incremental manner and used ontology-based reasoning only when pair-wise comparisons of specifications were performed. As many pair-wise comparisons may have to be made, the current algorithm's rationale is to construct a bigger KB involving C specifications and not just 2 as we expect that this will take less time than having to reason over $C - 1$ KBs of size 2 (if we assume that always the first specification is constant, i.e., the request). Moreover, we use incremental classification to construct the offer hierarchy as this might be deemed better than having to construct this hierarchy in a pair-wise manner. As such, we expect that this algorithm will be faster than the previous one.

Algorithm Core. In offer registration, for each X offers stored, we perform incremental classification over the KB and store the classification hierarchy in main memory. Rationale is again that it is better to incrementally do this every time a specific number of incoming offers is issued rather than running incremental classification on-demand for each incoming offer to be registered.

Matchmaking follows a similar rationale as in the previous algorithm. The sole exception lies on the fact that now the classification is more complete but also contains new offers (less than X) that have not yet been classified and are considered top-nodes. Due to the classification completeness, we also need to keep track of the nodes visited so as not to revisit them again. The matching process starts by matching the top-nodes in the classification hierarchy in chunks of C nodes each time (see *Rationale* paragraph). If a top-node is subsumed, we do not follow its descendants but just add them in the matching offers set. We also mark this node and its descendants as visited. Otherwise, we need to go down the top-node's tree to find matches again similarly to top-node matching.

Complexity Analysis. Offer registration is equivalent to the case of the incremental classification algorithm. Thus, its time complexity is: $O\left(L_{spec} * N + \sum_{Y=1}^{\frac{N}{X}} S_X^{Y*X}\right)$.

Request matching has 3 cases. In the best case, the request matches a left top-node in the 1st subsumption chunk taking: $O\left(L_{spec} + S_C\right)$. In the worst case, we must match the request with all nodes. This takes $O\left(L_{spec} + \frac{N}{C} * S_C\right)$, further reduced to $O\left(\frac{N}{C} * S_C\right)$. In the average case, we make the same assumptions as in previous algorithm. The time complexity is: $O\left(L_{spec} + \frac{N*\left(1-\frac{P}{2}\right)}{C} * S_C\right)$, further reduced to $O\left(\frac{N*\left(1-\frac{P}{2}\right)}{C} * S_C\right)$.

5 Experimental Evaluation

The experimental evaluation aims at comparing the proposed algorithms with the *subMIPMM* mixed-based one to identify cases that these algorithms prevail. This evaluation relied on the experimental framework in [6]. It also exploits the second real dataset from WS-Dream collection [10] and one randomly constructed in a controlled manner. The main comparison metric is average execution time for both registration and matchmaking. Accuracy has not been considered as all algorithms are perfect in this aspect [6] by completely realising the matchmaking metric of specification conformance [1]. In the following, we first shortly explain the way experiments have been performed and then present each experiment's results along with their respective analysis.

Please note that Pellet was used for ontology subsumption in the algorithms while the Ibex constraint solving framework (`www.ibex-lib.org/`) was exploited for constraint matching in *SubMIPMM*.

5.1 Experiment Set-Up

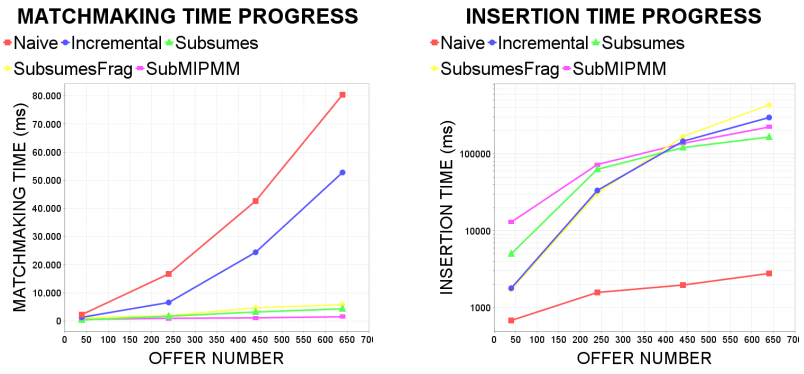
All experiments were performed in a laptop with a 64bit OS, a 6GB main memory and a multicore CPU of 2.4GHz frequency. For each experiment, we have conducted a series of steps to produce the respective average measurements of the algorithms considered. Each step maps to specific fixed or dynamic values of the control parameters and a series of 30 runs from which the average was calculated in order to alleviate for interferences at the OS level.

Real or randomised input was used in the experiments. In case of WS-Dream dataset, depending on the offer number (given as a control parameter value), we randomly selected an equal number of measurements from around 4500 available ones which were transformed into respective ontology-based offers mapping to the two main terms exploited, i.e., *response time* and *throughput*. The corresponding request was randomly selected again from the 4500 measurements. In case of the randomised dataset, the offers were randomly created based on current values of the control parameters. More details about this can be found in [6]. The respective randomised request was constructed again based on the control parameter values so as to match a specific percentage of offers.

5.2 1st Experiment

In this experiment, we exploited the randomised dataset and considered that half of the offers will be matched by each request issued. The number of offers was linearly increasing from 40 to 640 with a step of 100. The respective experiment results are visualised in Figs. 4a and 4b.

Concerning matching time, it is clear that *SubMIPMM* algorithm is the best, followed by *Subsumes*. *SubsumesFrag* comes next while in the end we have *Incremental* and *Naive*. These matchmaking results were not expected especially between the two ontology-based subsumes algorithms while *SubMIPMM* prevailance possibly indicates that there is a bound in the variable number always



(a) (b)

Fig. 4: Fig. (a) shows matching time results for 1st experiment while Fig. (b) shows registration time results for 1st experiment

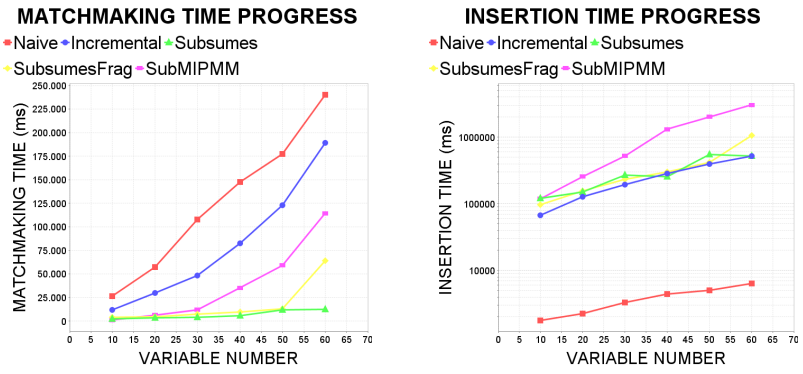
leading to constraint-based matchmaking being faster than ontology subsumption. Please note that there is a speedup with respect to incremental reasoning which is not great as the removal of a previous request and the addition of a new one (based on the way the experiment was conducted) in the existing KB requires performing subsumption over a great number of offers.

Concerning registration time, it was obvious that *Naive* will be the best while *subMIPMM* the last. However, the second expectation was not realised as there is a specific breakpoint in *SubsumesFrag* performance because incremental reasoning is not efficient due to the nature of specifications and subsumption's exponential complexity. The order change between *Subsumes* and *SubMIPMM* is due to the fact that the latter performs two (complex) constraint solvings per comparison in registration in contrast to just one for matchmaking while obviously the former performs just one classification per comparison constantly.

5.3 2nd Experiment

In this experiment, we exploited again the randomised dataset with almost similar control parameter values but: (a) the offer number is now constant (300) and (b) we linearly increase the QoS term number in the specifications from 10 to 60 with a step of 10. Our main goal is to show that as the QoS term number increases, the constraint number in each QoS specification also increases; as such the number of constraint problems to be solved by *subMIPMM* also increases. In this sense, we expect that there will be a specific bound on the QoS term number beyond which subsumption reasoning will be quicker than pair-wise constraint-based matching. Fig. 5 shows the experiment results.

Concerning matching time, the results are in accordance to the previous experiment ones for the same initial variable number. However, as soon as the variable number goes to 20, we clearly see that *SubMIPMM*'s performance gets



(a) (b)

Fig. 5: Fig. (a) shows matching time results for 2nd experiment while Fig. (b) shows registration time results for 2nd experiment

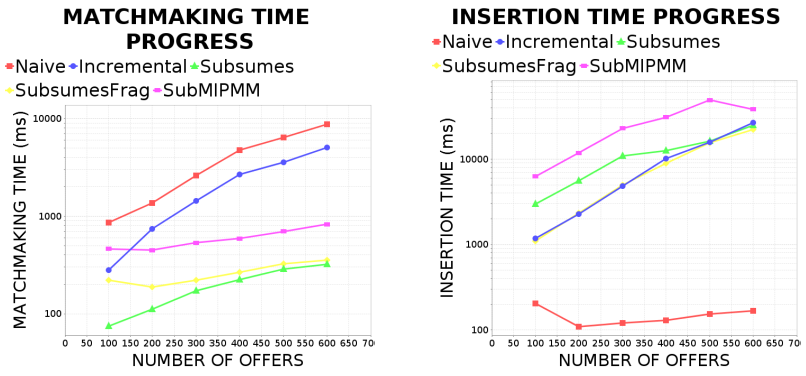
worse and less than that of the ontology-based subsumes algorithms whose performance order is not altered throughout the experiment. The order between *Naive* and *Incremental* is also not altered with respect to the previous experiment, something quite expected.

Concerning registration time, it is clear that *SubMIPMM* is the worst algorithm as it has to increasingly solve a much higher number of constraint problems per offer registration when the number of QoS variables and respective constraints increases. The order between the two ontology-based subsumes algorithms is almost the same which is evident also by the theoretical complexity analysis. As in the previous experiment, the order and performance of the rest of the ontology-based algorithms is not modified.

5.4 3rd Experiment

In this experiment, we exploit the real dataset and increase the offer number from 100 to 600. So, similar settings as in 1st experiment apply with two exceptions: (a) the QoS term number is 2 and not 10; (b) it is expected that the QoS offer number to be matched is small and thus much more work is expected for all subsumes-based algorithms. The main goal is to stress-test the algorithms in real situations and inspect whether the last algorithm can outperform the rest as it will have to perform less subsumption checking pieces of work. Figs. 6a and 6b visualise the respective results.

Concerning matching time, the results validate the complexity analysis as all subsumes algorithms exhibit a linear behaviour while the rest an exponential one. We also see a difference with respect to the 1st experiment results as a much better algorithm performance is exhibited. This can be possibly due to the fact that the variable number is less so each matchmaking piece of work takes less



(a) (b)

Fig. 6: Fig. (a) shows matching time results for 3rd experiment while Fig. (b) shows registration time results for 3rd experiment

time. In addition, we now see that *SubMIPMM* is worse than the ontology-based approaches from which *Subsumes* is still the best.

Concerning registration time, the results are expected based on our assumptions as *SubsumesFrag* is the best among all subsumes algorithms followed by *Subsumes* and then *SubMIPMM*. The behaviour of *SubsumesFrag* and *Incremental* coincides, as expected. Obviously, the *Naive* algorithm has constantly the best performance in all experiments according to this aspect.

Two main derivations must be highlighted from the above results: (a) a smart ontology-based approach can outperform a constraint-based one under real circumstances and (b) *Subsumes* seems to be the best algorithm in the long run for both registration and matchmaking – this can be seen from the breakpoint at 500 in the x-axis for registration beyond which this algorithm is better than *SubsumesFrag*. The latter also reveals the main weakness of even an incremental reasoner due to the nature of the specifications that it has to address and the exponential complexity in subsumption.

6 Conclusions

This paper has presented a pure ontology-based approach in QoS-based service matchmaking. This approach is realised by a naive and two clever algorithms which intelligently organise the service advertisement space. The latter two algorithms significantly outperfom the naive one in matching time and even compete with recent state-of-the-art QoS-based service matching algorithms. This is clearly shown in the randomised and realistic experimental evaluation where the cases in which our novel algorithms prevail are detected. Based on our propositions and findings, we showcase that a pure ontology-based approach when assorted with smart algorithms and techniques can really compete with other

QoS-based service matching approach types. So, we provide guidance to practitioners under which circumstances an ontology-based approach can be exploited.

Concerning future work, the following directions are planned. First, further investigation of new algorithms which more cleverly organise the advertisement space. Second, checking the modification of the normal subsumption reasoning process in order to cater for: (a) not requiring the modification of negatively monotonic QoS terms to positive ones and (b) for more cleverly matching QoS-based service specifications. Third, performing a more thorough evaluation with the state-of-the-art to detect additional cases where a pure ontology-based approach should be recommended. Fourth, coupling the novel approach proposed with a semantic functional matchmaker in order to realise a complete ontology-based service discovery system. Such coupling could also lead to cleverly and concurrently organising and matching the offer space according to both specification aspects to further speed up the overall matchmaking time.

Acknowledgments This research has received funding from the European Community's Framework Programme for Research and Innovation HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

1. Cortés, A.R., Martín-Díaz, O., Toro, A.D., Toro, M.: Improving the Automatic Procurement of Web Services Using Constraint Programming. Int. J. Cooperative Inf. Syst. 14(4), 439–468 (2005)
2. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases. pp. 372–383. VLDB Endowment, Toronto, Canada (2004)
3. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benbernou, S., Brandic, I., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. 46(1), 1 (2013)
4. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: ECOWS. pp. 265–274. IEEE Computer Society (2006)
5. Kritikos, K., Plexousakis, D.: Requirements for QoS-Based Web Service Description and Discovery. IEEE Transactions on Services Computing 2(4), 320–337 (2009)
6. Kritikos, K., Plexousakis, D.: Novel Optimal and Scalable Nonfunctional Service Matchmaking Techniques. IEEE T. Services Computing 7(4), 614–627 (2014)
7. Plebani, P., Pernici, B.: URBE: Web Service Retrieval Based on Similarity Evaluation. IEEE Transactions on Knowledge and Data Engineering 21(11), 1629–1642 (2009)
8. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), 51–53 (2007)
9. Sycara, K.P., Paolucci, M., Soudry, J., Srinivasan, N.: Dynamic Discovery and Coordination of Agent-Based Semantic Web Services. IEEE Internet Computing 8(3), 66–73 (2004)
10. Zhang, Y., Zheng, Z., Lyu, M.R.: WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services. In: ISSRE (2011)
11. Zhou, C., Chia, L.T., Lee, B.S.: DAML-QoS Ontology for Web Services. In: ICWS. p. 472. IEEE Computer Society (2004)