

Discovery of Optimized Web Service Configurations Using a Hybrid Semantic and Statistical Approach

Maciej Zaremba
Digital Enterprise
Research Institute (DERI)
National University of Ireland
Galway, Ireland
maciej.zaremba@deri.org

Jacek Migdal
Department of Mathematic,
C. S. and Mechanics
University of Warsaw
Warsaw, Poland
jacek@migdal.pl

Manfred Hauswirth
Digital Enterprise
Research Institute (DERI)
National University of Ireland
Galway, Ireland
manfred.hauswirth@deri.org

Abstract

We present a *Semantic Optimized Service Discovery (SemOSD)* approach capable of handling Web service search requests on a fine-grained level of detail where we augment semantic service descriptions with statistically built predictor functions. Our approach combines ontologies and mathematical functions built using statistical regression over previous Web service interactions. In the search requests we allow for arbitrary, independent and dependent constraints and user preferences expressed using objective functions. Our approach maps to standard Operational Research global optimization problem where algorithms of Simulated Annealing and Differential Evolution are used. It is capable of finding the optimal combination of service input and output parameters (a configuration) to a user request with rich preferences. Our approach is applied to an international package shipment scenario where real (Web) services are used and mined to create price prediction models. We show that the chosen regression method provides price prediction models of high accuracy and our approach supports expressive and complex search requests.

1 Introduction

Service discovery and late binding is at the heart of the Service Oriented Architecture (SOA) enabling *visibility* [8] between services, yet no comprehensive approach for describing and discovering services exists. Most of the service discovery approaches operate on abstract service descriptions specified either syntactically [4] or semantically [2, 9]. A main problem of existing service discovery approaches is their inability to operate on the fine-grained level of service descriptions. Discovery performed on the abstract service descriptions matches a user request on the class or category level where further details have to be manually determined by inspecting and communicating with the service. On the abstract level, the user has to search first by service category

(e.g., using keywords or ontology concepts and constraints) and then manually input required data to find out the individual service configuration. On the other hand, fine-grained service discovery operates on the detailed service description dynamically generated to the service request at hand.

A single service can have thousands of different configurations which depend on the user input. Enumerating all service configurations using standards like WS-Agreement [1] is not scalable, as the number of configurations grows exponentially with the number and range of input parameters. Output parameters like price, delivery time, QoS attributes are part of the service configuration and often can be determined only by communicating with the service. Service pricing policy is a particular example of information that service providers do not reveal and tend to provide only on an individual request basis. However, as more business is carried out online it is becoming feasible to mine service policies and to utilize them for discovering the most suitable service configurations.

We propose a service configuration discovery where dependencies between service input and output parameters are modeled using *predictor functions*. *Predictor functions* are created by machine learning techniques analyzing logs of the previous interactions with the service. Using statistical regression we build multi-dimensional piecewise linear functions and high-degree polynomials reflecting service input/output dependencies. *Predictor functions* reflect actual service capabilities what helps in semantically and accurately describing service. More importantly *predictors* are used in finding optimal service configurations.

Our search request model is specified as an objective function handling independent, dependent and arbitrary constraints between input and output parameters. Users can further refine their goals following optimization results and can inspect complex dependencies between inputs and out-

puts parameters using *predictor functions* plots. We formulate our problem of finding optimal service configurations as a task of searching for the global function minima in Operational Research (OR) using stochastic algorithm of Simulated Annealing [6] and genetic algorithm of Differential Evolution [10]. User's input/output parameters specified as preferences map into global function which minima is determined using aforementioned OR algorithms. We extend our previous work on service instance discovery [12] with a rich objective function-based preference model, multi-level constraint support and service (pricing) policy mining.

We present the application of our configuration discovery in an international package shipping scenario where real (Web) services of companies like USPS, TNT or FedEx were we utilized and mined. We provide an implementation of our discovery framework as an open source software¹. Our approach proves to be expressive and able to handle arbitrary combinations of input and output constraints in a reasonable time using high accuracy *predictor functions* as we show in the evaluation part.

2 Service Offer Configurability

Configurable services are services in which the output parameters (price, delivery time, insurance coverage, QoS, etc.) depends on the user's input parameters. By service configuration we understand the combination of input data provided by the user to the service and output data provided in response by the service, what leads to service instantiation. Some Web services are highly configurable [7] where both their input parameters P^I and output parameters P^O can vary significantly leading to a considerable number of possible service configurations. Each of the parameters in P^I and P^O can be of continuous (ranges of values) or discreet (enumeration of values) data type. Set of input parameters (P^I) is defined as the following:

$$P^I = P_1^I \times P_2^I \times \dots \times P_n^I, \quad n \in \{1, 2, 3, \dots\} \quad (1)$$

P_i specifies the allowed values for the given parameter which can be either enumeration of values (v_1, v_2, \dots, v_k) or ranges of values in case of continuous data types (set of value ranges with lower - v^{Lower} and upper v^{Upper} bounds).

$$P_i \in \{v_1, v_2, \dots, v_k\}_{Enum} \vee \quad (2)$$

$$P_i \in \{(v_1^{Lower}, v_1^{Upper}), \dots, (v_l^{Lower}, v_l^{Upper})\}_{Ranges} \quad (3)$$

P_{Inst}^I denotes a concrete binding of input parameters to allowed values following the enumeration and ranges constraints. Similarly, P_{Inst}^O denotes concrete output binding generated under P_{Inst}^I .

P^O parameters are defined similarly to P^I , however, for each output parameter there is a transformation function expressed in terms of input parameters. Service output P^O dependent on P^I is defined as the following:

$$P^O = P_1^O \times P_j^O \times \dots \times P_m^O \quad (4)$$

$$F_j^O : P^I \rightarrow P_j^O \quad (5)$$

F_j^O is a transformation function that provides the value of parameter P_j^O under given input or parts of input P^I . F_j^O reflects output parameters of the given service including business sensitive information like pricing policy. Typically, function F_j^O resides within the private IT space of a service provider and P_{Inst}^O is only available to the customer by invoking the service with its P_{Inst}^I .

We distinguish two types of invocable service methods relevant from the configuration perspective: **(1) Safe methods** do not have any real world effect. Their role is clearly informative and obtained information may constitute a service offer. Service quotation method providing price and delivery time is an example of a *safe method*. **(2) Real-world effect methods** perform actions changing the state of the real world. *Real-world effect method* is typically invoked once the user is aware of the service offering and wants to proceed with it. Purchasing products or approving money transfer are examples of methods having effect in the real world.

Invoking a service *safe method* gives the service requestor access to the P_{Inst}^O offered for the provided P_{Inst}^I , in some cases a *real-world effect method* may also provide access to P^O . It is sound from the business perspective to first provide the customer with the quote information (P^O in this case) and then to allow him/her to commit to the offer resulting in the promised real-world effects.

2.1 Requirements

We propose the following requirements on the process of service configuration discovery: **(R1) Service configuration description** should reflect the service structure including offered product categories, and involved input/output parameters. Considering highly configurable services, the effort required to describe service functionality by the domain expert should be minimized by utilizing information from previous service interactions. Service description should reflect real functionality and not only be built on top of existing textual or technical service descriptions. **(R2) User request (goal)** has to express detailed requirements over the service configuration allowing for complex preferences and flexibility. The user should be able to constrain both input and output parameters of the service. User should not be limited with expressing his constraints over service input and output parameters. If constraints over the given input or output parameter are not specified then the optimal value should be chosen. **(R3) Ontologies** should be used to express both goals and service descriptions. Goals and services are decoupled and links between them can be

¹<http://maczar.derri.ie/SemOSD/SemOSD-v05.zip>

established by using ontologies. Logic inference can be applied to deduce relationships between a goal and a service. **(R4) Scalability** of the discovery framework. Service configuration discovery aims at the further automation in the late binding process [11] therefore, services and their configurations should be discovered during runtime in reasonable time.

3 Service Configuration Discovery

3.1 Service Configuration Description

Services are described in terms of the hierarchy of the ontology concepts with attached constraints. The following are the main elements of service description: **(1) Service** is a top level entity. Service level constraints include constraints of all possible service configurations. The example of a service is package shipping having attached information of a maximum weight and package size that the service is able to handle. **(2) Categories** contain constraints that are common for the parts of the service functionality (e.g., category of international shipment or category of domestic shipment – each having different constraints in terms of handled origin and destination countries). **(3) Product Types** are the finest grain type of service offering. Product constraints are specific to the given product type (e.g., Priority Mail International, Express International). *Predictor functions* (described in more detail in Section 4.3) of output parameters are attached to *product types*.

For computational reasons we separate constraints into three main categories: (1) **independent input constraints** specify allowed and disallowed input parameters' ranges and enumerations (e.g., $1 < weight < 50kg$ or $currency \in \{USD, EUR\}$), (2) **dependent input/output constraints** specify dependencies between input and output parameters (e.g., $length + girth < 108$ or $\frac{price}{weight} < 4$) and finally (3) **arbitrary constraints** which specify any additional constraints including rules (e.g., 20% discount applies if product ordered within next 24 hours). Both (1) and (2) are specified as mathematical equalities and inequalities. We rewrite conditional expression of (3) into equalities in form of the mathematical piecewise functions:

$$f(x_1, x_2, \dots, x_n) = \begin{cases} value_1, & \text{if } rule_1(x_1, \dots, x_n) \\ \dots & \\ value_n, & \text{if } rule_n(x_1, \dots, x_n) \end{cases} \quad (6)$$

The example mentioned above can be rewritten as:

$$discount(orderTime) = \begin{cases} 20, & \text{if } orderTime \leq 24 \\ 0 & \end{cases} \quad (7)$$

Rewritten conditional variables of (3) like *discount* variable can be used in any other equalities or inequalities of (1) and (2). The provided service model satisfies both R1 and R3 as it is able to properly reflect service structure and related constraints.

3.2 Search Request

For finding an optimal service configuration the user should specify input P^I and output P^O parameters that should be minimized or maximized. The most obvious choice is minimizing the service price, however, there are other parameters (e.g., delivery time, insurance coverage) that the user could minimize or maximize. User preferences are expressed in an objective function which assigns weights to inputs and outputs to be optimized. An objective function will only specify parameters that a user cares about and our approach finds optimal values for the parameters that are not constrained by the user. We define the objective function as the following:

$$F_{Obj}(P^I, P^O) = coeff_1 * F(P^I) + \dots + \quad (8)$$

$$coeff_j * F(P^I, P^O) + \dots + coeff_n * F(P^O) \quad (9)$$

Coefficients specify the weights of P^I and P^O that should be minimized and maximized. *coeff* has a negative value in case of parameter maximization and positive value for minimization. Parameters used in objective functions are subject to same type of independent, dependent and arbitrary constraints as in the case of services. The provided goal model allows for complex constraints and arbitrary preferences complying with R2. The input and output parameters to be optimized are expressed in the ontology language which satisfies R3.

3.3 Global Optimization

We formulate our problem of finding the optimal service configuration as a numerical global optimization problem which deals with optimization of a set of functions to the given criteria under certain constraints [10]. We use an objective function to specify user preferences, output parameters are expressed using *predictor functions* and independent, dependent and arbitrary constraints specify the solution feasibility region as a set of equalities and inequalities.

Simulated Annealing (SA) is one of the most popular probabilistic (stochastic) methods [6] of locating good approximation of a function global minimum being subject to various non-linear constraints in large search space. The initial state is randomly chosen and is changed in every step with the neighbouring state picked with a probability dependent on the global parameter T (called temperature). In the beginning T is high and choosing neighbouring states is almost random, as the algorithms progresses T gradually decreases to zero.

Differential Evaluation (DE) is a genetic and heuristic algorithm for finding the global minimum of a multi-dimensional, constrained function [10]. It starts by sampling an objective function at a number of points which satisfy the provided complex constraints. Random vectors are selected from the generated vector population and new difference vectors (perturbations) are calculated and mutated.

Results of the mutations are compared with existing vectors and those which gain a lower value for the objective function are used for further mutations. This operation is performed until the convergence criteria is satisfied and no further improvements can be made.

4 Service Configuration Lifecycle

Four lifecycle steps, presented on Figure 1, are necessary to apply our discovery mechanism over any domain.

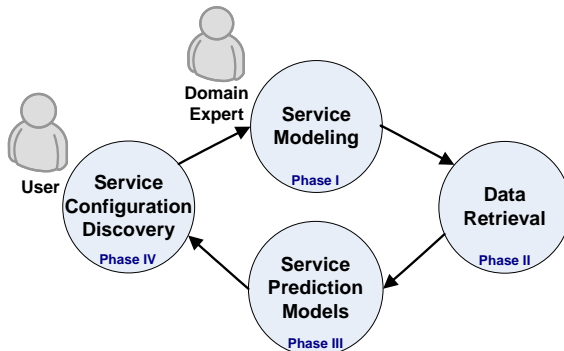


Figure 1. SemOSD Lifecycle

4.1 Service Modeling

In an ideal world information provided in service technical descriptions (WSDL, XML Scheme) and textual documents describing service functionality suffice for enhancing service with a semantic annotations. In reality, the technical descriptions, like WSDL documents, do not provide enough expressivity for describing details of service functionality. A domain expert has to provide a semantic description of a service following the available documentations (WSDL, XML Schema, textual documents).

The domain expert must: **(1) Select common parameters.** The first step is to find existing services in the given domain (e.g., international shipping companies, insurance companies). Result set has to be reviewed in order to select common attributes (e.g., weight, price), units of parameters (e.g. EUR, USD, kg, cm) and determine attribute granularity (only destination country is taken into account or also the postal code). Some attributes are marked as required (like source and destination country). **(2) Create service descriptions.** Ontology of products has to be created as shown in Section 3.1 following technical and textual documentation of a service. For each product type, parameters constraints are provided to narrow product attribute space. It has to be specified from which input parameters *predictor functions* should be created. Independent, dependent and arbitrary constraints between service parameters can be also specified and attached to service, categories and product types. **(3) Specify Web service grounding.** Mapping common attributes from domain ontologies to XML messages specific to the given Web service takes place via service grounding. **(4) Choose regression and optimization**

method. In some cases it is feasible to choose a domain specific regression method and optimization. A selection of regression and optimization methods is already provided as well as generic default methods. **(5) Perform model verification.** Technical and textual documentation may contain outdated information and only by executing the service it can be verified what parameters are returned, their ranges and possible enumerations. Domain expert assumptions can be verified by running a number of test interactions with the service.

4.2 Data Retrieval

The main source of data that can be used for building *predictor functions* are the previous interactions with the service. There are two complementary approaches for gathering the data: **(1) Requested service quotations** are run in the initial phase to gather sufficient equally distributed sampling points that reflect service input and output dependencies. **(2) Provided service quotations** are obtained in the course of the discovery engine use. The source of the records could be: results verification, feedback from client, third-party, etc..

Typically *requested service quotations* are more useful as they provide a controlled access to service quotations and it can be guaranteed that the entire input range will be covered and equally distributed. On the other hand, the *provided service quotations* do not require to run service sampling invocations which can be seen as obtrusive by some service provides. Typically a fewer equally distributed data records of *requested service quotations* allow to build more accurate prediction model than in case of heavily clustered *provided service quotations*. The whole transaction record is stored in its original version as unit conversions (e.g., currency exchange rates) change over the time.

4.3 Service Prediction Models

4.3.1 Continuous parameters

The general problem of building accurate *predictor functions* boils down to the fact that there is typically a few magnitudes less data points available than all of the possible input combinations. There is no generic, all-purpose, accurate method for multi-dimensional function approximation based on sampling points (observations). Although not public, the service pricing policy functions are typically not very complex. Our evaluation showed that international shipping companies tend to calculate price using additive piecewise linear function with multiple breaking points dependent on weight and requested insurance.

A domain expert can select a few, most influential input parameters as function arguments. Complexity of *predictor functions* depend on selected parameters. Input that has negligible influence on modeled output parameter can be omitted in order to simplify *predictor function*. In shipping scenario, package size does not have direct influence over

price for some of the shippers. Constraint on the maximum package size supported by the shipper has to be specified. Shipment price is derived from weight and requested insurance irrespectively from specified package size.

One of the regression methods has proven to be especially accurate. It assumes that the *predictor function* arguments are independent from each other and that overall output function is built from the additive piecewise linear functions of independent parameters. Output parameters like a service price are defined as a baseline value with number of additive and independent piecewise functions. It is not trivial to rebuilt such a *predictor function* as its component piecewise functions typically have multiple breaking points.

Using this premise, derivative can be calculated separately for each of the independent piecewise functions. Having all of the derivatives and chosen sample point (in fact there are many sample points available, but only one needed) we can transform a derivative of a function into a function.

More formally, we want to build a *predictor function*:

$$f(x_1, x_2, \dots, x_n) \quad (10)$$

Having points:

$$S = \{(x_{1,1}, x_{1,2}, \dots, x_{1,n}, f(x_{1,1}, x_{1,2}, \dots, x_{1,n}))\} \quad (11)$$

To do this for each of the argument x_i , we select all pairs:

$$(x_{k,1}, x_{k,2}, \dots, x_{k,n}, f(x_{k,1}, x_{k,2}, \dots, x_{k,n})) \quad (12)$$

$$(y_{p,1}, y_{p,2}, \dots, y_{p,n}, f(y_{p,1}, y_{p,2}, \dots, y_{p,n})) \quad (13)$$

Where:

$$\forall_{j=1}^n j \neq i \Rightarrow x_{k,j} = y_{p,j} \quad (14)$$

For each of the argument x_i we have set of triples $(x_{k,i}, y_{p,i}, \Delta_{k,p,f})$. Then using those triples a following function is created a result of solving/approximating a set of linear equations:

$$g_i(x_i) = \frac{d}{dx_i} f'(x_1, x_2, \dots, x_n) \quad (15)$$

$$\text{Where } \forall_{\text{triple}} \int_{x_{k,i}}^{y_{p,i}} g_i(x) dx \approx \Delta_{k,p,f} \quad (16)$$

Having all of the $g_i(x)$ we select a sample point $(y_1, y_2, \dots, y_n, f(y_1, y_2, \dots, y_n))$ and $f(x_1, x_2, \dots, x_n)$ is a *predictor function*:

$$f(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_n) + \sum_{i=1}^n \int_{y_i}^{x_i} g_i(x_i) dx \quad (17)$$

To model services we use the Web Services Modelling Language (WSML) [3] ontology language. Generated *predictor functions* are specified using Wolfram's Mathematica

language². Listing 1 shows 3D piecewise functions created using linear regression specified in a WSML ontology as Mathematica three-dimensional piecewise function.

```

1 ...
2 instance predictorPrice memberOf {Piecewise3DFunction, Predictor}
3 hasInput hasValue {"weight", "insuranceValue" }
4 hasOutput hasValue "price"
5 hasFunction hasValue "funcprice[weight., insuranceValue.]:=
6 Piecewise[{{1500*weight-1500, 1<=weight<5},
7 {1425*weight-1400, 5<=weight<7},
8 {950*weight+1450, 7<=weight<11},
9 {950*weight+1925, 11<=weight<16},
10 {950*weight+2400, 16<=weight<19},
11 {975*weight+1875, 19<=weight<26}}]+
12 Piecewise[{{0.0*insuranceValue,
13 40<=insuranceValue && insuranceValue<2044}}]"
14 ...

```

Listing 1. Price Predictor in WSML

4.3.2 Discrete parameters

Although static regression is reasonable for continuous parameters like weight or requested insurance, it is not the best choice for discrete types such as country codes or cities. The discrete types are usually limited to a much smaller space of possible values than continuous types. In our approach a separate *predictor function* is constructed for each discrete value. All *predictor functions* that appear to be the same are clustered in one group (based on a few random points). The groups are continuously verified. If two points appear with the same input arguments, but with different output value, the group is split. The groups are useful, since they are common and they drastically reduce the number of required quotes.

For example, in the shipment scenario, available (source \Rightarrow destination) tuples: (USA \Rightarrow Ireland), (USA \Rightarrow UK), (USA \Rightarrow Germany), can be merged into groups: (USA \Rightarrow (Ireland, UK, Germany)), (Ireland \Rightarrow UK, UK \Rightarrow Ireland).

4.4 Service Configuration Discovery

In the discovery process user request is matched against existing services and a search for an optimum configuration is carried out within each service. The independent input parameter constraints of service and goal are intersected, and in case there is common space, a search for an optimal service configuration can proceed. If constraints over inputs and outputs are not specified we assume that the user expects optimal values of unspecified input and output parameters.

Considering the main elements of our goal being expressed using constraints and objective functions we can distinguish two phases as shown on Figure 2 of service configuration discovery: Phase I - **independent input parameters filtering** by intersecting goal and service, category and product type independent input parameters, Phase II - **global optimization** where the optimal solution to an

²<http://www.wolfram.com>

objective function is sought by considering complex dependent and arbitrary constraints over input and output parameters. Phase I is computationally inexpensive as determining the common space of independent input parameters is very fast. This phase selects *product types* that are likely to contain a solution to the problem at hand. Phase II is applied to the *product types* selected in Phase I and global optimization of objective function with checking the satisfiability of complex independent, dependent and arbitrary constraints is performed. Global optimization algorithms first determine whether a solution exists within the specified constraints and then search for the objective function minima or maxima.

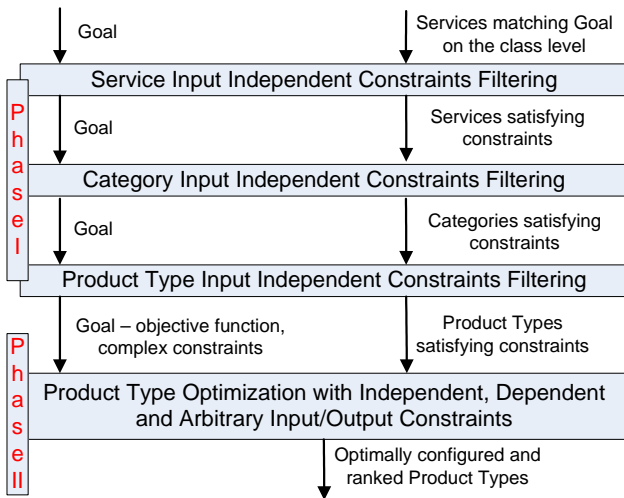


Figure 2. SemOSD Filtering and Optimization

We use Simulated Annealing and Differential Evaluation methods (Section 3.3) for finding optimal service configurations. Both methods use the same type of constraints expressed as equalities and inequalities of input and output service parameters operating on objective functions. This makes switching between them straightforward. The resulting output includes the minimal value for the provided objective function what is relevant for ranking various *product types* within and across services, and the optimal values of input parameters. Optimization is performed on each service product type that has passed the *independent input parameters filtering*. The result of the optimization process is a ranking of different *product types* normalized to the human natural scale of 0 (worst) – 10 (best).

5 International Package Shipping Scenario

Ordering an international package shipment is a standard example of online services. It was chosen as an example domain due to the availability of real services. Companies do not necessarily want to have their offers easily compared with competitors requiring users to use their own Web interfaces (e.g., AJAX Web forms) which hinders the automated

use of services by third-party software like our discovery engine. However, shipping companies tend to be more open making their services available online as Web services (e.g., FedEx, DHL), REST services (e.g., USPS) or HTML forms (e.g., TNT). In case where there was no Web service access possible (e.g., TNT), it was necessary to provide Web service adapters to enable unified access to different shipping service using REST or to employ advanced screen scraping software like Screen Scraper³.

Restriction	User input
Constraints	$P_{from}^I = USA, P_{to}^I = Ireland$ $200 \leq P_{insurance}^I$ $8 \leq P_{weight}^I \leq 10$ $1 \leq P_{deliveryTime}^O \leq 3$ $(\frac{P_{price}^O}{P_{weight}^I}) \leq 5$
Objective function	$F_{Obj}(P_{price}^O, P_{weight}^I, P_{insurance}^I) =$ $P_{price}^O - P_{weight}^I * 5 - P_{insurance}^I * 10^{-5}$

Table 1. User Goal Example - Formal Notation

A typical user request in this scenario is sending a parcel. It is assumed that the client knows the source and destination address, but he/she might only estimate the parcel dimensions, weight and declared insurance value. A shipping company responds only when all of the input values are given and responds with a list of possible offers. Each of the package type has two output parameters, delivery time and price. The customer has some restrictions (like package weight range) and preferences. The preferences specify which values should be minimized/maximized and how important it is for the client. The importance of parameters and whether they should be minimized or maximized is expressed using coefficients as explained in Section 3.2. Table 1 shows an example of expressive shipment goal.

The service configuration discovery presented in this scenario address two problems: unknown pricing policy and large number of possible configurations. Shipping companies mostly provide quotation only per individual request as it is not feasible to list all of the possible configurations. Moreover, many requests which appear intuitive to the end user are not currently available (e.g. “I would like to ship a 5 kilo package for at most 50 euro, as fast as possible”).

6 Evaluation

We have run a series of tests within ranges of input values allowed by shipping services. Sampling quotes were obtained by invoking Web services (including our Web service adapters) of companies like TNT, USPS and FedEx. Input sampling points were not generated completely randomly, samples were run along single input parameter dimension at the time. For instance, we have run service sampling along fixed source and target location, fixed insurance

³<http://www.screen-scraper.com>

value, but randomly sampled different weights within specified range. In another run, we have harvested number of sampling points along fixed weight, but flexible insurance value. Such a strategy allowed us to build more accurate *predictor functions* and more easily observe commonalities when comparing to samples run randomly along all input dimensions. Sampling quotes that were not used in the process of creating *predictor function* were utilized to verify the accuracy of the predictions.

We used the standard deviation, worst absolute deviation and accuracy measures to evaluate the quality of the created *predictor functions*. N defines the number of sampling points that were used to verify quality of *predictor functions* and were not used to build these *predictor functions*. $|D|$ is an absolute deviation between observed value x_i (value received from the service) and expected value $E(X_i)$ predicted by the created predictor function. $|D_{[\%]}|$ is the absolute deviation $|D|$ normalized to 0 (worst)–100 (best) percentage range. Standard deviation σ is expressed using absolute deviation. Accuracy $Acc_{[\%]}$ of a given predictor is specified as a mean of $|D_{[\%]}|$.

$$|D| = |x_i - E(x_i)| \quad (18)$$

$$|D_{[\%]}| = 100 - \frac{|D| * 100}{x_i} \quad (19)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (|D|)^2} \quad (20)$$

$$Acc_{[\%]} = \frac{1}{N} \sum_{i=1}^N |D_{[\%]}| \quad (21)$$

Table 2 presents the obtained results. Tests were run in 50 iterations and the average accuracy depending on the number of points randomly chosen from the set of 300 sampling points was calculated. Average accuracy reaches 95% for more than 30 sampling points for TNT, what we consider to provide sufficient prediction accuracy. It has to be noted that there is no publicly documented information available that would allow for building such prediction models for shippers like TNT and FedEx. USPS provide price information in the form of human readable documents, but only for domestic shipping. REST and Web interfaces are provided for USPS international shipment price quotation. Plots showing worst and average accuracy of USPS and TNT *predictor functions* are shown on Figure 3.

Table 3 presents time results for the example goals. $Goal_1$ specifies a request of discovering the cheapest shipment between two countries of package between a 10 and 15 kg, with insurance between 100 and 400 EUR. The objective function specifies that a user is able to pay extra 1 EUR per each kg more, and 10 EUR per each day less. $Goal_2$ additionally specifies that delivery time should be equal or less than 6 days and that the price per kg should be below 20 EUR. The tests have been carried out on a

computer with Intel Core 2 Duo 2.0GHz and 3GB of RAM. Implementation of both Simulated Annealing and Differential Evolution provided by Wolfram’s Mathematica has been applied to evaluate the objective functions on number of the *Product Types*. As presented in the Table 3, the time required per each optimization is close to 150 ms ir-respectively of the goal and *predictor function* complexity. Checking *independent input parameters constraints* generates negligible time overhead, and most of the time is consumed in *Product Type* global optimization. As the time evaluation shows, our approach scales linearly satisfying scalability requirement R4.

Used Points P	σ	Worst $ D $	Worst $ D_{[\%]} $	$Acc_{[\%]}$
10	10.02	20.68	64.12 %	90.80 %
15	7.82	16.64	71.93 %	92.94 %
20	6.68	14.37	74.97 %	94.00 %
25	6.19	13.47	77.51 %	94.54 %
30	5.06	11.60	80.42 %	95.56 %
40	4.79	10.89	81.61 %	95.79 %
50	4.55	10.29	82.23 %	95.99 %
80	3.54	7.99	87.11 %	97.12 %

Table 2. Regression Quality Results - TNT

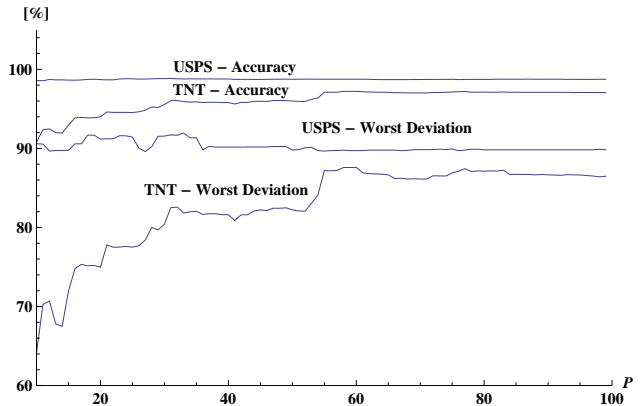


Figure 3. Price Predictor Quality

Goal	Product Types	Optimized P. Types	Time [ms]	Time[ms] per optimization
$Goal_1$	19	7	1094	156
$Goal_1$	95	35	5140	147
$Goal_1$	380	140	23516	168
$Goal_2$	19	12	2125	177
$Goal_2$	190	120	18813	156
$Goal_2$	380	240	41734	173

Table 3. SemOSD Time Tests

7 Related Work

The most related recent work to our discovery and ranking approach of highly configurable services is Lamparter et al. [7] where domain expert is expected to specify attribute functions, which in case of complex pricing policies

of multi-breaking points functions might be a quite daunting task. Our discovery approach caters for complex constraints specified both in terms of service input and output data. Lamparter et al. approach handles only manually modeled price as an output parameter, while in our case approximated *predictor functions* are built for all output parameters of the service.

The work of Etzioni et. al [5] on mining airline price policies is also related to our work. Similarly to our case they have gathered and analyzed a number of price quotations to derive price policy model. In the case of airline fares, the time factor is one of the most important parameters influencing the price while in case of international shipment prices are fairly stable and there are only periodic pricing policy updates. However, the purpose of two approaches is different, goal of Etzioni's approach is to find optimal time to purchase airline ticket focusing on one dimensional optimization parameter (price). Goal of our approach is a discovery of optimal service configuration following preferences specified in objective function complying with complex input/output constraints.

Paolucci et. al [9] propose functional semantic discovery using state transitions between pre- and post-conditions which operates on the high level of service abstraction while our approach targets the fine-grained aspects of service functionality. In fact, standard static pre- and post-conditions discovery can be used in the filtering process before applying our optimal configuration discovery in the final phase.

8 Conclusions

We have presented a generic framework, implementation and evaluation of a service configuration discovery that can be adapted to any kind of domain of highly configurable services. The main problem that our method faces is public availability of real, complex and configurable Web services. Companies typically provide custom-made Web interfaces and are reluctant to provide public access to their business via Web service interfaces. Providing Web service adapters with employed screen scraping is merely a workaround in resolving the issue of price quotation access.

We built multi-attribute functions by applying regression to the data obtained from past interactions with services, which results in less human labour and better reflects changes of underlying policies. Optimization of service configurations is carried out in terms of minimizing the objective function with complex dependencies between input and output parameters by using algorithms of Simulated Annealing and Differential Evolution. Our approach is suited to configurable services with complex dependencies between input and output parameters (e.g., like pricing policies) that are not disclosed by service providers. Thus it is difficult for the user to make a fully informed decision and find an optimal service offering. As shown in the eval-

uation, our discovery approach provides high accuracy and scales linearly.

Based on these promising results, we also plan to investigate further applicability of our approach in the domains of highly configurable services like car insurance and financial services.

9 Acknowledgments

The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and the European Union project SUPER (FP6-026850).

References

- [1] A. Andrieux, K. Czajkowski, A. Dan, K. K. Pruyne, et al. Web Services Agreement Specification (WS-Agreement), Grid Resource Allocation Agreement Protocol (GRAAP) WG Specification, 2007.
- [2] J. Colgrave, R. Akkiraju, and R. Goodwin. External Matching in UDDI. In *International Conference on Web Services (ICWS)*, 2004.
- [3] J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. The Web Service Modeling Language WSM: An Overview. In *European Semantic Web Conference (ESWC)*, 2006.
- [4] A. Dogac, Y. Tambag, P. Pembedioglu, S. Pektas, et al. An ebXML infrastructure implementation through UDDI registries and RosettaNet PIPs. In *International Conference on Management of Data (SIGMOD)*, 2002.
- [5] O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To Buy or Not to Buy: Mining Airfare Data to Minimize Ticket Purchase Price. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.
- [6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science Magazine*, 220(4598), 1983.
- [7] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm. Preference-based Selection of Highly Configurable Web Services. In *International Conference on the World Wide Web (WWW)*, 2007.
- [8] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz. Reference Model for Service Oriented Architecture 1.0, OASIS SOA-RM Committee Specification, 2006.
- [9] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference (ISWC)*, 2002.
- [10] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., 2005.
- [11] T. Vitvar, A. Mocan, M. Kerrigan, M. Zaremba, M. Zaremba, M. Moran, et al. Semantically-enabled Service Oriented Architecture: Concepts, Technology and Application. *Service Oriented Computing and Applications*, 1(2), 2007.
- [12] M. Zaremba, T. Vitvar, and M. Moran. Towards Optimized Data Fetching for Service Discovery. In *European Conference on Web Services (ECOWS)*, 2007.