Combining self-organisation, context-awareness and semantic reasoning: the case of resource discovery in opportunistic networks

Graeme Stevenson, Juan Ye, and Simon Dobson
School of Computer Science
University of St Andrews, UK
graeme.stevenson@st-andrews.ac.uk

Danilo Pianini, Sara Montagna, and Mirko Viroli
Alma Mater Studiorum
Università di Bologna, Italy
danilo.pianini@unibo.it

ABSTRACT

The increasing prevalence of networked devices brings ever more opportunities for delivering content and services to users that result from situated interactions between computational devices in their surrounding environment. Resource discovery, a vital component in this process, becomes challenging in such an open, dynamic and distributed setting. Building on earlier work that outlined a novel semantics-based approach to resource discovery in such environments, this paper provides a general solution to incorporating application-specific contextual factors into the resource discovery process, and proposes a mechanism to support the runtime evolution of resource discovery tasks in a mobile setting.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Self-organisation, Semantic Matching, Bio-inspired, Resource Discovery, Context-awareness

1. INTRODUCTION

Embedded sensing and computational technologies in everyday devices, such as smartphones and public information displays, are becoming ever more advanced. The promise of opportunistically networking such devices offers as yet untapped potential in supporting the the dynamic provision of services based on ad-hoc, spontaneous arrangements of devices, services, and content across an infrastructure with few central points of control.

We have proposed SAPERE [1] as an architecture for orchestrating interactions across such *service ecosystems*, with data lifecycle and communications regulated by bio-inspired mechanisms with robustness, scalability and adaptability characteristics [2].

SAC'13 March 18-22, 2013, Coimbra, Portugal.

Supporting expressive resource discovery atop this unstructured, highly-dynamic substrate presents a challenge, with the combination of semantic reasoning, context-awareness and self-organisation techniques offering intriguing possibilities: semantic matching provides a flexible framework for quantitatively assessing the suitability of resources to a given request; context-awareness may account for application quality of service concerns that are important, but orthogonal to the matching process, and self-organisation techniques provide a means of adapting to changing environmental conditions—the arrival and removal or resources, the mobility of interacting agents, and the ever evolving network topology.

To illustrate the utility of this approach, we consider a smartphone application for an exhibition centre (for example, Tokyo's Makuhari Messe, which contains eleven exhibition halls and has a footprint of $210,000m^2$), which offers routing information and exhibit suggestions based on a user's profiled interests. These techniques support the development of exploration strategies to: *i)* select the exhibit most suited to the user's interests, *ii)* select preferential exhibits while accounting for factors such as their distance from the user's position or crowdedness along the route, *iii)* route the user to an exhibition hall containing many exhibits of interest, but not necessarily the most interesting if it is considered spatially isolated, *iv)* use a combination of the above strategies, for example, to guide the user to a particular exhibition hall and then route them to the exhibits inside the hall in order of interest or least crowded.

In earlier work [3] we outlined a novel approach to resource discovery using ontological resource descriptions and semantic matching to *i)* provide distributed semantics-based resource discovery that selectively routes responses to requests based on their match-degrees, and *ii)* design a framework within which match-degrees may be dynamically influenced contextual factors, such as distance. Here, we extend this work in three directions:

- An improved algorithm provides a generalised solution to the problem of application-driven contextualisation of semantic match results. This encompasses not only distance (previously hardcoded) but other application-specific contextual factors extrinsic to the resource matching process such as crowdedness, communication latency, and node stability.
- An bio-inspired pattern additional to those previously identified, *decay*, provides a mechanism to support dynamic lifecycle management. It underpins the self-organisation of resource discovery in the ecosystem over time in response to the addition, removal, and mobility of resources.
- A quantitative evaluation through simulation characterises the performance of our approach.

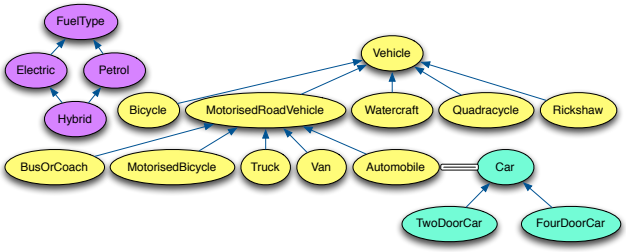

Figure 1: An partial illustration of concepts in the Vehicle Sales Ontology [7], an equivalence relation from one term to another concept hierarchy, and a snippet from a fuel type ontology.

In Section 2 we discuss semantic matching and its realisation within the SAPERE framework. In Section 3 we describe a self-organising, semantic resource discovery process atop the SAPERE model. In Section 4 we provide implementation details, followed by an evaluation of the approach in Section 5. We briefly discuss related work in Section 6 and present concluding remarks in Section 7.

2. SEMANTIC MATCHING IN SAPERE

SAPERE [1] envisions a fully-decentralised pervasive computing ecosystem, where services are delivered based on opportunistic interactions between resources spatially distributed in an environment. This section introduces semantic matching, and describes how it is employed to aid resource discovery within SAPERE.

2.1 Semantic Matching

A matchmaking process takes a request and a set of resource descriptions as input, and outputs the set of resources that satisfy the request. Many matchmakers adopt simple syntactic schemes; for example, named interfaces or predefined categories [5], or sets of attribute-value pairs to which string and numeric comparisons can be applied [6]. Such approaches are limited by an inability to compare semantically equivalent but syntactically different concepts or handle approximation. To illustrate, consider the concept hierarchy depicted in Figure 1, based on the Vehicle Sales Ontology [7]. Here, the concepts `Automobile` and `Car` may be considered as equivalent terms, or `TwoDoorCar` as an approximate match to a `FourDoorCar`. Semantic analysis supports the incorporation of such matches where they would otherwise be discounted.

A semantic matching process has three parts: a model for describing requests and resources, a scheme for measuring the semantic similarity of concepts, and an algorithm for evaluating the match.

Modelling Resources and Requests.

A request expresses the intersection of a number of properties, each named concept or existential restriction belonging to a resource. Using Description Logics (DL) notation [8], the following describes a request for two door, electric cars manufactured after 2009, and three advertisements for candidate vehicles to be matched against this request.

R: $TwoDoorCar \sqcap \exists modelDate \geq 2009 \sqcap \exists fuelType = Electric$
$A1$: $Automobile \sqcap modelDate = 2009 \sqcap fuelType = Hybrid$
$A2$: $TwoDoorCar \sqcap modelDate = 2011 \sqcap fuelType = Petrol$
$A3$: $Van \sqcap modelDate = 2011 \sqcap fuelType = Electric$
$A4$: $Bicycle \sqcap modelDate = 2007$

None of the four advertisements fully satisfies the request: there is insufficient information to determine if $A1$ is a $TwoDoorCar$; $A2$ differs in the requested fuel type; although electric, $A3$ is a Van; and $A4$ satisfies no request aspects. However, by inspection we can see that some of the advertisements more closely relate to the request than the others. We say that an advertisement is *compatible* with a request if the intersection of their descriptions is satisfiable.

Quantifying Semantic Similarity.

Paolucci et al. [9] and Li et al. [10] introduce ordered degrees of matching to categorise the semantic compatibility between a request (R) and an advertisement (A), of which there are five: *exact*, *plugin*, *subsume*, *intersection*, and *disjoint*. Bandara et al. [11] extend this model with scoring mechanisms to support differentiation of matches falling within the same category. Descriptions of the match-degrees and formulas for scoring taxonomically related concepts are shown in Table 1.

Applying this mechanism to the concept hierarchy in Figure 1 allows us to quantify the similarity between terms in our earlier example. For example, the degree of similarity between $Automobile$ and $TwoDoorCar$ is 0.75 (plugin), the degree of similarity between $TwoDoorCar$ and Van is 0.5 (intersection), and the degree of similarity between $Hybrid$ and $Electric$ is 1 (subsumption).

Executing the Semantic Match.

A semantic matching algorithm conflates all the terms in a request and resource description and may be realised in many ways. The pseudocode for one possible implementation is given below:

function MATCH(R, A)
  if $A \sqsubseteq R$ **then** ▷ R subsumes, or equivalent to A
    return 1
  end if
  if R and A are atomic concepts or literals **then**
    return `SIMILARITY(`R`,` A`)`
  end if
  $score \leftarrow 0$
  for all R_i in R **do** ▷ Composite concept/requirement
    $score \leftarrow score$ + `MAX(MATCH(`R_i`,` $A_{1..m}$`))`
  end for
  return $score/|R|$ ▷ Return average score of composite
end function

The `MATCH` algorithm takes two variables R and A as input, which correspond to the request and advertisement or some subset of their terms respectively. The function first checks if R subsumes A; if so, the algorithm terminates with a similarity score of 1. If R and A refer to atomic named concepts or literals, the similarity score is calculated by a call to the `SIMILARITY` function, which resolves the comparison to a score depending on how the concepts are related. If not, R and A are composites consisting of named concepts or existential restrictions. This case is handled by recursively decomposing R into its constituent parts, $R_{1..n}$, and averaging the maximum score for each R_i, when compared with corresponding parts of A, that is, $A_{1..m}$. Applying this algorithm to the above example yields the scores: $A1$=.92, $A2$=.83, $A3$=.83, and $A4$=.08.

2.2 Realisation in SAPERE

The SAPERE architecture supports pervasive services bound to the locality and context in which they execute by reifying data and events in the regions of space where they pertain, and by promoting interactions based on proximity [1]. Agents, acting on behalf of user applications and available services, express their state as "Live

Match Degree	Description	DL Notation	Similarity Score (R, A)
Exact	The request and advertisement are equivalent concepts.	$R \equiv A$	1
Subsume	The request expresses a more general concept than the advertisement.	$A \sqsubseteq R$	1
Plugin	The request expresses a more specific concept than the advertisement.	$R \sqsubseteq A$	$\frac{\lvert S(A)\rvert}{\lvert S(R)\rvert}$
Intersection	The intersection of the request and advertisement is satisfiable.	$\neg(R \sqcap A \sqsubseteq \bot)$	$\frac{\lvert S(A) \sqcap S(R)\rvert}{\lvert S(R)\rvert}$
Disjoint	The request cannot be satisfied by the advertisement.	$R \sqcap A \sqsubseteq \bot$	0

Table 1: Match-degrees and their semantic scores, defined using the similarity metric proposed by Skoutas et al. [4]. $S(X)$ denotes the set of super-concepts of concept X in its defining ontology.

Semantic Annotations" (LSA) that continuously reflect the state of their associated components (live), which is connected to the domain in which such information is produced, interpreted and manipulated (semantic). LSAs are reified in a networked, distributed space (an "LSA-space") acting as the *fabric* of the ecosystem.

LSAs have a unique identifier (ID), and a content that includes the information the agent wants to manifest. They are realised as an RDF-like [12] set of triples that consist of a subject (an ID), a predicate (the property name, a URI) and an object (the assigned value, a literal, URI, or locally scoped identifier). By adopting a notation resembling N3 [13], an LSA is represented as "`id p v; id q w1 w2 w3;`" where `id` is the ID, property `p` is assigned to value `v`, and property `q` is assigned to values `w1`, `w2`, and `w3`.

Aspects of autonomous adaptation are achieved following the natural inspiration [14] through designing *self-organising* system rules called *eco-laws* that – by executing actions upon a small set of *co-located* LSAs – make global properties emerge.

Eco-laws are structured as chemical-resembling rules [15] of the kind "`P+..+P --> Q+..+Q SideConditions`". Elements `P` and `Q` are patterns of LSAs, expressed like N3's LSAs with the following changes: *(i)* values are either strings or URIs; *(ii)* in place of each element of a triple one can use a variable `?V` (matching any value); *(iii)* variable constraints are lifted to an unordered sequence of SPARQL [16] `SideConditions`, which are either "`FILTER` (*exp*)" or "`BIND` (*exp* `as ?V`)"; *(iv)* each predicate can be prepended by symbols +, – and =, the former assumed by default — respectively meaning that the triples with this object should exist, should not exist, should be the only that exists for that subject and predicate. Additionally, we sometimes use an expression of the kind "`?LSA: clones ?LSA2`", meaning that `?LSA` should have the same content as `?LSA2` plus any following constraints.

The semantics of an eco-law *reaction* is that of consuming *reactant LSAs* based on left-hand side patterns and producing a set of *product LSAs* based on right-hand side patterns. Eco-laws obey a numeric transformation rate `r`—a Markovian rate in a continuous-time Markov chain system. If omitted, the rate is assumed to be infinite, that is, the eco-law is executed with "as soon as possible" semantics. Through their agents, applications and services perceive the world through transformations affecting their LSAs.

The complete framework is realised as a lightweight middleware that reifies LSAs in the form of semantic tuples, to be dynamically stored and updated in a system of spatially-situated tuple spaces spread over the devices of the network. The eco-laws governing the ecosystem apply locally in all network nodes [17, 18].

3. DISTRIBUTED RESOURCE DISCOVERY

We now clarify how standard self-organisation patterns can be augmented with semantic reasoning so as to support a decentralised approach to resource discovery in pervasive computing applications.

3.1 Self-organisation Patterns

Fernandez-Marquez et al. [2] provide a catalogue of mechanisms for self-organisation in terms of modular and reusable design patterns. This section briefly reviews those patters that form our interaction building blocks.

Spreading.

The Spreading pattern progressively sends information over the system from one node to its neighbourhood, iteratively, so as to make it available globally by using only local interactions. In our framework it is supported by the following eco-law:

```
?A :val ?V; :diff ?F; :loc ?L
-->
?A + ?B :#clones ?A; :val =?W; :loc ="*"; :prev = ?L
?BIND (:exec2(?F,?V) AS ?W)
```

Aggregation.

The Aggregation pattern reduces the amount of information in the system, typically by summarising data disseminated by agents over time, e.g., by spreading. It is supported by:

```
?A :src ?S; :aggr ?F; :val ?V + ?B :src ?S; :val ?W
-->
?A :val =?Z :
?BIND (:exec3(?F,?V,?W) AS ?Z)
```

Decay.

The Decay pattern incrementally ages information in the system. A monotonically decreasing function is applied to a numeric decay value over time, with the information removed upon the counter reaching zero. Decay-based Aggregation supports the overwriting of old information from a data source with the more recent. Decay is supported by the following eco-laws:

```
?A :decay ?D :dfunc ?F;
-->
?A :decay ?N;
?BIND (:exec4(?F,?D) AS ?N)

?A :decay 0;
-->
```

Gradient.

The Gradient pattern is a composition of the Spreading and Aggregation patterns, where information about the sender distance and direction is propagated across an extent of the network. An LSA can become source of a gradient by specifying initial distance 0, a diffusion function incrementing the distance value, and an aggregation function retaining the less traveled LSA among competing gradient LSAs from the same source [19]. The Decay pattern supports *dynamic* gradients by discarding old information as a gradient source moves within the network.

Chemotaxis.

The Chemotaxis pattern provides a mechanism to route information towards the source of a gradient via the shortest accessible

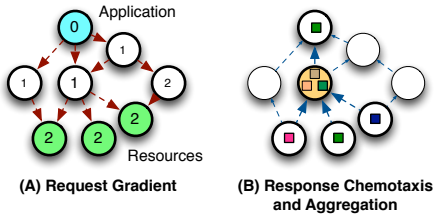

Figure 2: Resource discovery interactions: (A) a request gradient is spread, and (B) replies ascend the gradient and aggregate.

path. In our framework, movement of LSAs towards a gradient source is achieved by diffusing in the direction indicated by property `:prev` as created by the Spreading eco-law.

3.2 Self-organising, Context- and Semantics-aware Resource Discovery

Building upon the above concepts, we now introduce the key steps in our approach to self-organising, context- and semantics-aware resource discovery, which are illustrated in Figure 2.

Establish the resource request.

An application searching for a resource publishes a `request` gradient LSA carrying information about the source R of the request: spreading and aggregation eco-laws will iteratively fire in random establishing a gradient data structure, with horizon H (hop distance counter), and in which each LSA has a pointer to the node in which it was created, which is also the node indicating direction to the source following shortest path.

Resolve matches and publish reply.

Where a gradient LSA containing a request is co-located with an annotation describing a resource (or service) S, we apply semantic matching, as described in Section 2 to resolve their compatibility to a match-degree. This is realised by a matching eco-law that creates a `reply` LSA pointing towards the request and the service, and with the value MD representing the match-degree (greater than 0).

Route match information to the requestor.

The reply LSA is created such that it diffuses by copying towards the request source; namely, the chemotaxis pattern is applied to deliver a remote representation of the resource annotation matched to the request source. This is continually applied to maintain a live link between the two points in the network: the request site and service site. This ends after the request gradient expires, for example, after the request is explicitly removed. This allows the process to update and self-heal under conditions where *(i)* an intermediate node used for routing fails, *(ii)* a compatible resource appears within the gradient, *(iii)* the requestor or resources are mobile within the network.

Augmenting matches with application context.

As the reply LSA is routed, aggregation may be employed to modify, either positively or negatively, the reported match-degree, using information external to the reply. This is achieved by applying an aggregation function to any LSA directed towards a request source that accounts for contextual factors (*any information that can be used to characterise the situation of entities that are considered relevant to the interaction* [20]) specified by the application; and with such information readily available for consumption through its expression in an LSA. For example, $\langle S_0, 0.9, 0.75\rangle$ (resource, original-match-degree, match-degree) might represent a reply LSA that has been updated to account for the distance an LSA has travelled, or the presence of physically crowded regions along the response path.

Aggregate results en route by competition.

Where a request is matched to multiple, distributed resources, the algorithm returns the most suitable to the requestor. For example, in the case where there are two co-located response annotations of the form $\langle S_0, 0.9, 0.75\rangle$ and $\langle S_1, 0.8, 0.8\rangle$ (resource, original-match-degree, match-degree), the latter will be propagated while the former will be dropped. Combining this with the previous mechanism provides a contextual-semantic approach whereby a resource may be retained that, although it has worse match-degree when considered in isolation, better accounts for a requestor's contextual requirements.

Aggregate results en route by collaboration.

An alternative to the above approach is to augment the standard semantic-matching process by making a resource more, or less, attractive depending on some contextual relation to similarly matched resources. For instance, in applications where resources represent physical points of interest (POIs), we may consider a small region of the space that includes many POIs more favourable than another region containing the POI with highest match-degree. This is achieved by an aggregation function that "joins" two replies instead of selecting one of them. For example, response annotations of the form $\langle S_0, 1, 0.9\rangle$ and $\langle S_1, 2, 0.75\rangle$ (resource, distance, match-degree), can be subject to joining as they represent spatially close services. Hence, they are aggregated into a single annotation $\langle S_0 + S_1, 1.5, 0.85\rangle$, in which distance and match-degree summarise the original ones. As before, application specified contextual factors may readily be accounted for.

Receipt of replies.

The requestor will eventually receive aggregated replies, providing information about (summary) distance, (summary) match-degree, resource provider (or resources provider), and direction to reach the service. Such information can then be used, for example, to steer the requestor agent towards a selected resource or selected resources, as developed in [19].

Dynamic self-healing.

The evolution of ecosystem states over time is supported by the Decay pattern, which is employed to periodically remove older information while newer information is concurrently refreshed. The periodic update of request gradients and reassertion of responses accounts for the requestor moving closer to or further away from resources, for resources disappearing or arriving, and for the corresponding impact these changes have on the aggregation applied to the match degree results in transit.

4. ECO-LAWS TO SUPPORT MATCHING

In this section we present eco-laws that support the contextual-semantic management of resource discovery, describing their behaviour and the functions they encapsulate. For the sake of space we will not provide the general-purpose eco-laws supporting the spreading, aggregation, gradient, and chemotaxis self-organisation patterns, for they are discussed elsewhere [19]. We instead focus on the four eco-laws that incorporate the key strategies described in previous section, namely, the generation of replies along with a match-degree, their contextualisation at each node they tran-

Resource discovery with match-degree and spatial information

```
%[MATCH] When a request and service match, a reply LSA is generated to be diffused towards requestor
?REQ :type :request; :content ?R; :source ?SRC + ?SER :type :service; :content ?S
--?Rate-->
?REQ + ?SER +
?REP :#clones ?REQ; :type = :reply; :service ?SER; :content ?S; :original_match_degree ?MD; :match_degree ?MD;
     :ctxed "false"; sos:direction ?SRC; sos:switcher :ctxed; sos:type sos:ascend
BIND(:semanticMatchDegree(?R, ?S) AS ?MD)
FILTER(?MD > 0)
BIND(:replyRate() AS ?Rate)

%[CTX] A reply contextualises, and is accordingly propagated one-step
?REP :type : reply; :ctxed "false"; :ctxprop ?P; :ctxtype ?T; :ctxfun ?F; ?P ?V
?CTX :type ?T; ?P ?V2
--?Rate-->
?REP :ctxed = "true"; ?P = ?W +
?CTX
BIND (:exec(?F,?V,?V2) AS ?W)
BIND(:diffRate() AS ?Rate)

% [AGGREGATE-COOP] Joins information from two replies belonging to the same service cluser
?A1 :type :reply; :ctxed "true"; :request ?R; :service ?S1; :ctxprop ?P; ?P ?V1; :match_degree ?MD1;
    :coop_pred ?CP; :coop_md ?CM; coop_ctx ?CC +
?A2 :type :reply; :ctxed "true"; :request ?R; :service ?S2; :ctxprop ?P; ?P ?V2; :match_degree ?MD2 +
--->
?A1 :service ?S; ?P = ?V; :match_degree = ?MD
FILTER(:exec(?CP,?V1,?V2)) .                % The two services should "cooperate"
BIND (:exec(?CC,?V1,?V2) AS ?V) .           % Let V be the aggregated contextualisation
BIND (:exec(?CM,?MD1,?MD2) AS ?MD) .        % Let MD be the aggregated match-degree
BIND (:union(?S1,?S2) AS ?S) .              % Let S the aggregated service description

% [AGGREGATE-CHOOSE] Selects between two replies originating within different service clusters
?A1 :type :reply; :ctxed "true"; :request ?R; :service ?S1; :ctxprop ?P; ?P ?V1; :match_degree ?MD1;
    :comp_pred ?CP; :coop_sel ?CS +
?A2 :type :reply; :ctxed "true"; :request ?R; :service ?S2; :ctxprop ?P; ?P ?V2; :match_degree ?MD2 +
-->
?A :service ?S1; ?P ?V; :match_degree ?MD1
FILTER(:exec(?CP,?V1,?V2)) .                % The two services should "compete"
FILTER(:exec(?CS,?V1,?MD1,?V2,?MD2)) .      % Service A1 has a stronger match-context pair
```

Figure 3: Eco-laws for handling request-reply match, and for distributed aggregation.

sit across, and the competition-/collaboration-based aggregation as they are routed towards the requestor. They are presented in Figure 3, and described in turn. These eco-laws make use of a default (omitted) name-space corresponding to the ontology of resource discovery, the `sos` namespace incorporating the concepts relating to self-organisation patterns, and the `sapere` namespace incorporating all the general concepts of pervasive service ecosystems.

Eco-law `[MATCH]` creates the reply. It takes a request LSA `REQ` and a service LSA `SER` located in the same node, and creates the reply LSA `REP` such that: it clones `REQ`, has type `:reply`, points (via so-called bonds) to the service LSAs (`?SER`) and stores its content (`?S`), it is also of type `sos:ascend` relative to the gradient generated by `SRC`, and finally specifies properties `:ctxed` as flag to state whether contextualisation already took place. It also describes the match-degree `?MD`, computed by function `:semanticMatchDegree` over service content `?S` and request content `?R`, implemented by the algorithm shown in Section II—only match degrees greater than 0 are realised as a reply. Note that this eco-law is reapplied over time so as to continuously support the chemotaxis pattern as already mentioned, as such, external function (constant) `:replyRate` is used to extract the application rate `?R`.

Eco-law `[CTX]` ensures a reply LSA `?REP` is properly contextualised before being diffused one step. It seeks another LSA of type `?T` having property `?P` assigned to a value `?V2`, and accordingly changes `?REP` by switching the value of `:ctxed` flag and updating `?P` to the result of applying function `?F` to `?V2` and previous value `?V`. Switching such a flag causes the diffusion eco-law to move this LSA to a neighbour in the direction that ascends the request gradient, and to switch the flag back to false—so that eco-law `[CTX]` can be applied again. Before diffusion takes place, however, aggregations should occur according to the following eco-laws.

Eco-law `[AGGREGATE-COOP]` joins two reply LSAs perceived as belonging to a unique cluster of services (e.g., related points of interest). It takes two reply LSAs for the same requestor `?R`, whose contextualisation values make them be perceived as "near" by requestor-specified predicate `:cooperate`, and creates a new reply LSA (overwriting one of the two originals) in which the service indication is obtained by joining `?S1` and `?S2` by external function `:join`, contextualisation value `?V` is computed by function `?CC`, and match-degree is computed by function `?CM`.

The implementation of such functions strictly depends on the kind of contextualisation an application requires. In our simulations in the next section we shall use an estimated physical distance as a contextual property, and as such we consider the following function (to be compared with other approaches in the next activities of this research): `:coop_pred` simply checking that "distances" from the two resource are below a given threshold; `:coop_ctx` the weighted mean of the two distances based on match-degree, namely, $(?V1 * ?MD1 + ?V2 * ?MD2)/(?MD1 + ?MD2)$; and `coop_md` implements a t-conorm (a function to perform a "fuzzy union" [21]), and in particular the so-called product logic t-conorm $?MD1 + ?MD2 - ?MD1 * ?MD2$. Finally, `:join` simply appends the two arguments yielding the list of all service identifiers.

Eco-law `[AGGREGATE-CHOOSE]` selects one from two reply LSAs whenever they are perceived as belonging to different clusters of services. Similarly to the previous law, it takes two reply LSAs for the same requestor `?R`, whose contextualisation values

are such that they are not perceived as "near" by external function ?CP, and selects the one with the stronger semantic-spatial situation (discarding the other). This is computed by function ?CS. In our reference case study: :comp_pred is exactly the opposite of :coop_pred, while :comp_sel checks whether $(?MD1 * k/(?V1 + k)) \geq (?MD2 * k/(?V2 + k))$ for some fixed parameter k. This compares the two match-degrees, but penalises the one having greater distance (e.g., when $V = k$ the actual match-degree is halved).

5. EVALUATION OF APPROACH

As a proof of concept we use simulation conducted using ALCHEMIST [22], a prototype simulator extending the typical engine of a stochastic simulator for chemical reactions with the ability to express structured reactions (where chemicals can have a tuple-like structure and reactions apply by matching) and structure the system as a network of mobile nodes. Other simulators, such as the multi-agent based Repast [23], could be used, however, ALCHEMIST has been chosen due to its goal of bridging the gap between simulation code and the actual system specification in terms of eco-laws.

The simulated scenario aims to provide early validation of the model and functions (matching and aggregator operators) proposed in Section 4 to select the best service among those matching a request, and investigate the performance gain due to the use of our distributed aggregation techniques.

As a first step, we measured the overhead advantages due to the use of a distributed form of aggregation, both in space and in bandwidth, which we estimate by considering the number of reply LSAs that stabilise in the distributed set of node as a request is issued. We simulated a bi-dimensional environment populated with an infrastructural grid of 11x11 nodes in a rectangular structure. A request is initially injected from a boundary position. In each simulation we randomly placed N (ranging from 1 to 100) resources in the environment, each with a match degree randomly chosen between 0 and 1. The request generates a gradient, and all resources eventually send a trail of LSAs back towards the requestor. We compared the case in which no aggregation of such replies is performed, with the case in which as soon as two reply LSAs are co-located they compete such that only one of them is retained: it can hence be easily expected that a smaller number of LSAs will ultimately be diffused around. Note that the results of this simulation are independent of the aggregation behaviour. Results are reported in Figure 4, each showing an average value out of 50 runs. They show that while the overhead in a standard approach clearly grows linearly with the number of matching resources, distributed aggregation makes the number of involved LSAs grow much slowly, generally $2/3$ less for the particular case we considered—of course, different topologies can lead to different levels of improvement.

As a second example we show the advantage of using a cooperative approach during aggregation, which makes clusters of resources be preferable with respect to isolated resources. The reason why this is considered useful is that frequently (for example, think of a general shopping scenario), the first matching resource reached is not the right one to pick: a new resource might be discovered from there—depending on the application scenario, this may happen if the resource is no longer available, or the application user is not satisfied despite the match and wants to quickly find a nearby alternative. In that case a new request has to be issued: if the user already reached a cluster of matching resources, then it will more likely find another match nearby. To simulate the advantages of this behaviour we define a *fault probability* p that, as one resource is selected from the requestor in a node n, another one needs to be searched because of dissatisfaction. Our reference environment is

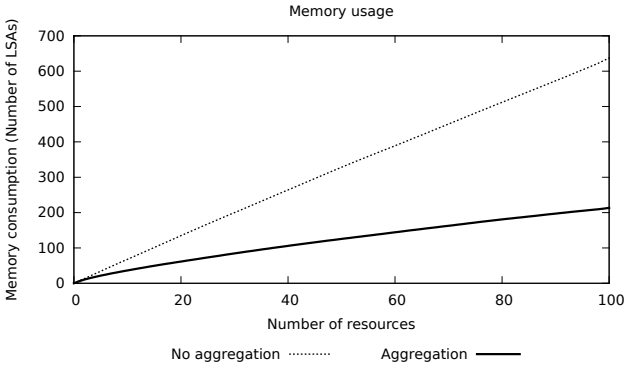

Figure 4: Number of reply LSAs depending on the number of matching resources.

the same as previous case, but we added 100 resources on one side of the rectangular area, to simulate a non-homogeneous displacement of resources that can take advantage of our management of clustering.

Figure 5 reports results, each considering a different value of p: the former measures average distance travelled (considering that many resources are generally to be reached), the latter measures the overall result quality, which is computed by the same formula $k * M/(D + k)$ we used to compare competing replies, that is, promoting resources which both are good matches and are near to the requestor. Both charts show that, even if the environment has no specific clusters, there is a significant gain in the quality of replies.

6. RELATED WORK

As described in [24], applications of coordination models and languages – and especially space-based ones – are inevitably entering the realm of self-organisation, where complexity of interactions becomes the key to make desired properties appear by emergence. Given the intrinsic difficulty of *designing emergence*, many approaches mimic nature-inspired techniques to organise and evolve data annotations spread in the system according to specified rules. To this end, several natural metaphors can be exploited, borrowing from physics, chemistry, biology, or social systems [14].

SwarmLinda [25] is a middleware that exploits the idea of the collective intelligence, displayed by swarms of ants, for guiding agents in charge of tuple storage and efficient tuple retrieval. Tuples are handled as sort of pheromones or items that ants (agents) relocate in order to improve overall efficiency. TOTA (Tuples On The Air) [26] is a tuple-based middleware supporting field-based coordination for pervasive-computing applications. In TOTA each tuple, when inserted into a node of the network, is equipped with a content (the tuple data), a diffusion rule (the policy by which the tuple has to be cloned and diffused around) and a maintenance rule (the policy whereby the tuple should evolve due to events or time elapsing). The *evolving tuples* model, presented in [27], extends traditional Linda tuple spaces with the goal of supporting resource discovery in a pervasive system, relying on ideas inspired to TOTA. The extension allows tuples to evolve so to be context-aware and able to adapt to environmental changes. Evolution is embedded in tuples by adding, to each field of the tuple, a name and a formula that specifies the field behaviour over time. Formulas support if-then-else constructs and arithmetic and boolean operators. Secondly a new operation evolve() is introduced in tuple space: it is responsible for applying formulas to tuples using context infor-

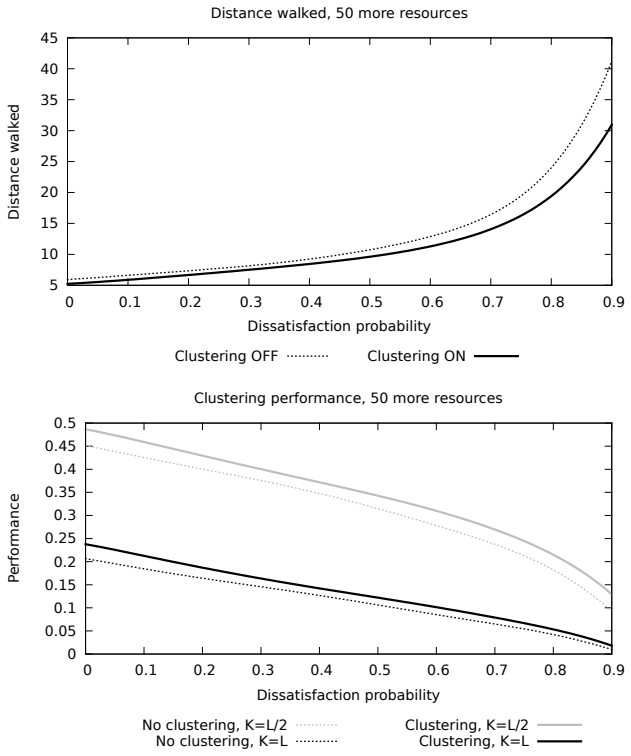

Figure 5: (top) Average distance travelled over fault probability, and (bottom) Average match-distance value over fault probability.

mation.

Although several approaches study semantic technique for tuple spaces [28, 29, 30, 31], the key idea of the work we present here is its ability to properly combining the openness supported by the use of semantic technologies with the emergence and adaptivity provided by the use of self-organisation.

In pervasive environments the diversity of service offerings gives rise to the need to balance the flexible expressivity of resource descriptions with an ability to compensate for the consequential difficulty in correctly constructing requests beyond the syntactic confines of approaches such as Jini [5] and SLP [6] that rely on interfaces and keyword matching.

Semantic Web technologies, founded on the use of shared, formal, and extensible vocabularies, provide useful constructs and reasoning frameworks that can support the expression of requests and resources involving syntactically different representations of semantically equivalent concepts. Chakraborty et al. [32] propose a peer-to-peer resource discovery protocol that employs the concept and property hierarchies of OWL [33] descriptions. Paolucci et al. [9] and Li et al. [10] introduce ordered degrees of matching, while in addition to the scoring mechanism described above, Bandara et al. [11] provide an OWL based matching algorithm that extends the basic scheme we present with support for property weights and required and optional features.

Beyond the semantic distance measurement we use here [4], approaches to quantifying the similarity between two concepts include: calculating the path length from their closest shared ancestor to to the root node [34] and counting the number of links between two concepts in a hierarchy [35]. For non ontologically related concepts, scoring based on the similarity of textual descriptions [36] and exploiting the frequency of concept occurrences in a given text corpus [37] has been employed.

To the best of our knowledge our approach is the first to use the semantic match-degree as a means of shaping the communications between network nodes, and to incorporate extrinsic contextual features into the discovery process in such a way that resource selection emerges as an active property of the bio-inspired communication process, as contrasted to *post hoc* analysis and filtering.

7. CONCLUSION

This paper presents extensions to a semantic-aware resource discovery mechanism, built on top of an ecosystem of opportunistically networked devices, in which the match-degree between requests and resources directly affects how the biologically inspired patterns that control interactions between devices are applied. The proposed extensions account for network evolution over time and provide a general approach for incorporating application-specific contextual factors that are extrinsic to the core matching problem. The result is a context- and semantics-aware algorithm for resource selection that self-organises in response to mobility and the arrival and departure of resources. A proof-of-concept evaluation validates our approach, and demonstrates the performance gain due to the use of distributed aggregation techniques in our approach.

Future works will be devoted to a more systematic evaluation, with more statistically significant results, along many different dimensions concerning also self-healing to topological changes, dynamism of the set of available resources, and focussing on real-life scenarios.

Acknowledgement

This work has been supported by the EU FP7 project "SAPERE - Self-aware Pervasive Service Ecosystems" under contract No. 256873.

8. REFERENCES

[1] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. D. M. Serugendo, M. Risoldi, A. Tchao, S Dobson, G. Stevenson, J. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally. Self-aware pervasive service ecosystems. *Procedia Computer Science*, 7:197–199, December 2011.

[2] J. L. Fernandez-Marquez, G. Di Marzo Serugendo, S. Montagna, M. Viroli, and J. L. Arcos. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing*, pages 1–25, 2012.

[3] G. Stevenson, M. Viroli, J. Ye, S. Montagna, and S. Dobson. Self-organising semantic resource discovery for pervasive systems. In *1st International Workshop on Adaptive Service Ecosystems: Natural and Socially Inspired Solutions*, pages 47–52, Lyon, France, September 2012.

[4] D. Skoutas, A. Simitsis, and T. K. Sellis. A ranking mechanism for Semantic Web service discovery. In *IEEE SCW*, pages 41–48, 2007.

[5] J. Waldo. *The Jini Specifications*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[6] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608, 1999.

[7] M. Hepp. The vehicle sales ontology. `http://www.heppnetz.de/ontologies/vso/ns`.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic*

Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA, 2003.

[9] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 333–347, London, UK, 2002. Springer.

[10] L. Li and I. Horrocks. A software framework for matchmaking based on Semantic Web technology. In *Proceedings of the 12th international conference on World Wide Web*, pages 331–339, New York, NY, USA, 2003.

[11] A. Bandara, T. Payne, D. de Roure, N. Gibbins, and T. Lewis. Semantic resource matching for pervasive environments: The approach and its evaluation. Technical report, University of Southampton, 2008.

[12] E. Miller and F. Manola. RDF primer. W3C recommendation, W3C, February 2004.

[13] T. Berners-Lee and D. Connolly. Notation3 (N3): A readable RDF syntax. W3C team submission, W3C, March 2011.

[14] F. Zambonelli and M. Viroli. A survey on nature-inspired metaphors for pervasive service ecosystems. *International Journal of Pervasive Computing and Communications*, 7(3):186–204, 2011.

[15] J. P. Banâtre and T. Priol. Chemical programming of future service-oriented architectures. *JSW*, 4(7):738–746, 2009.

[16] A. Seaborne and S. Harris. SPARQL 1.1 query. W3C working draft, W3C, October 2009.

[17] M. Viroli, F. Zambonelli, G. Stevenson, and S. Dobson. From SOA to pervasive service ecosystems: an approach based on semantic web technologies. In Javier Cubo and Guadalupe Ortiz, editors, *Adaptive Web Services for Modular and Reusable Software Development: Tactics and Solution*. IGI Global, 2012.

[18] M. Viroli and G. Stevenson. On the space-time situation of pervasive service ecosystems. In *Workshop on Spatial Computing*, Valencia, Spain, June 2012.

[19] M. Viroli, D. Pianini, S. Montagna, and G. Stevenson. Pervasive ecosystems: a coordination model based on semantic chemistry. In *Proceedings of SAC 2012*, Riva del Garda, TN, Italy, 26-30 March 2012. ACM.

[20] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction*, 16(2):97–166, December 2001.

[21] F. Bobillo and U. Straccia. Reasoning with the finitely many-valued Łukasiewicz fuzzy Description Logic. *Information Sciences*, 181(4):758 – 778, 2011.

[22] D. Pianini, S. Montagna, and M. Viroli. A chemical inspired simulation framework for pervasive services ecosystems. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 675–682, Szczecin, Poland, 18-21 September 2011. IEEE Press.

[23] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16(1):1–25, January 2006.

[24] A. Omicini and M. Viroli. Coordination models and languages: From parallel computing to self-organisation. *The Knowledge Engineering Review*, 26(1):53–59, March 2011.

[25] R. Tolksdorf and R. Menezes. Using swarm intelligence in Linda systems. In Andrea Omicini, Paolo Petta, and Jeremy Pitt, editors, *Engineering Societies in the Agents World IV*, volume 3071 of *Lecture Notes in Computer Science*, pages 519–519. Springer Berlin / Heidelberg, 2004.

[26] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications: The TOTA approach. *ACM Transactions on Software Engineering Methodology*, 18(4):1–56, 2009.

[27] D. Stovall and C. Julien. Resource discovery with evolving tuples. In *Proceedings of the international workshop on engineering of software services for pervasive environments*, ESSPE '07, pages 1–10, Dubrovnik, Croatia, 2007.

[28] L. J. B. Nixon, E. Simperl, R. Krummenacher, and F. Martin-recuerda. Tuplespace-based computing for the semantic web: A survey of the state-of-the-art. *Knowledge Engineering Review*, 23(2):181–212, 2008.

[29] D. Fensel. Triple-space computing: Semantic web services based on persistent publication of information. In *Intelligence in Communication Systems, IFIP International Conference*, pages 43–53, November 2004.

[30] M. Harasic, A. Augustin, P. Obermeier, and R. Tolksdorf. RDFswarms: selforganized distributed RDF triple store. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1339–1340. ACM, 2010.

[31] E. Nardini, M. Viroli, and E. Panzavolta. Coordination in open and dynamic environments with TuCSoN semantic tuple centres. In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, volume III, pages 2037–2044, Sierre, Switzerland, 22–26 March 2010. ACM.

[32] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin. Toward distributed service discovery in pervasive computing environments. *IEEE Transactions on Mobile Computing*, 5:97–112, 2006.

[33] M. Krötzsch, P. F. Patel-Schneider, S. Rudolph, P. Hitzler, and B. Parsia. OWL 2 web ontology language primer. Technical report, W3C, October 2009.

[34] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *ACL '94*, pages 133–138, Stroudsburg, PA, USA, 1994.

[35] C. Leacock, G. A. Miller, and M. Chodorow. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165, March 1998.

[36] M. Klusch and F. Kaufer. WSMO-MX: A hybrid Semantic Web service matchmaker. *Web Intelligence and Agent Systems*, 7(1):23–42, January 2009.

[37] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI'95*, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.