Self-organising Semantic Resource Discovery for Pervasive Systems

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

Mirko Viroli and Sara Montagna
Alma Mater Studiorum
Università di Bologna, Italy
Email: mirko.viroli@unibo.it

***Abstract*—The pervasive computing vision encompasses scenarios where services are delivered to users as a result of opportunistic encounters between their personal devices and computational resources embedded in their surrounding environment. The decentralised and dynamic nature of such environments complicates service provision, providing no setting for a conventional orchestrator to manage the resource discovery process. This paper proposes a novel approach to resource discovery, employing nature-inspired patterns to manage the search for and retrieval of information across a dynamic arrangement of devices. We show how the results of fuzzy matching based on semantic resource descriptions can be incorporated at the pattern level to route only the best matched resources to a requestor, and how application context extrinsic to the matching algorithm may augment this process.**

I. Introduction

Pervasive computing envisions the dynamic provision of services based on ad-hoc, spontaneous arrangements of devices, agents, and content across an infrastructure with few central points of control. Having previously proposed bio-inspired mechanisms with attractive robustness, scalability and adaptability characteristics as a communications layer upon which interactions between such arrangements of devices can be based [1], we develop here an approach to resource discovery using ontological resource descriptions deployed atop this substrate. A semantic matching scheme that resolves comparisons between requests and resources to a *match-degree* p, $p \in [0, 1]$ forms the groundwork for our two main contributions:

- A mechanism for distributed semantics-based resource discovery that selectively routes responses to requests based on their evaluated match-degrees, and self-adapts to environmental changes such as the addition, removal, and mobility of resources and the computational nodes that host them.
- A framework within which match-degrees may be dynamically influenced positively or negatively by application specific factors that are extrinsic to particular requests or resource descriptions. For example, the distance between the requestor and resource, the prevalence of similar resources in an surrounding area, or QoS metrics such as power consumption, latency, as error rate.

Incorporating fuzzy semantic reasoning within a self-organising communication framework provides a powerful mechanism for adapting pervasive systems' delivery of information based on dynamic context that benefits from the smooth integration of existing reasoning techniques.

Section II discuss the principles of semantic matching and its implementation inside the SAPERE framework. Section III describes the process of self-organising, semantics-based resource discovery atop the SAPERE model. In section IV we provide implementation details and a proof of concept via simulation. Finally, we present concluding remarks in section V.

II. Semantic Matching and SAPERE

SAPERE [1] envisions a fully-decentralised pervasive computing environment, where services are delivered based on opportunistic, nature-inspired interactions between resources distributed across an environment. In this section we introduce semantic matching, and describe how we employ it as part of a resource discovery process within SAPERE.

A. Semantic Matching

A matchmaking process takes a request and a set of resource descriptions (or advertisements) as input, and outputs the set of resources that satisfy the request. Many matchmakers adopt simple schemas; for example, named interfaces or predefined service categories [3], or sets of attribute-value pairs to which string and numeric comparisons can be applied [4]. Such approaches cannot compare semantically equivalent but syntactically different concepts or handle approximation — for example, `16:9` and `Widescreen` being equivalent aspect ratios, or `Red` as an approximate match to the colour `Scarlet`. Semantic analysis supports the incorporation of such matches where they would otherwise be discounted.

Additional benefits of semantic-based matchmaking over syntax-based alternatives include the flexible comparison of requests and resource descriptions expressed at different levels of abstraction (e.g., `ModeOfTransport` and `Bus`); automatic, rather than explicit, classification of resources through specifying the necessary and sufficient features required for a resource to fit a categorisation; and consistency checking of resource descriptions their defining ontologies [5].

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 I: Match-degrees and their associated semantic scores. $S(X)$ denotes the set of superclasses of concept X (Skoutas at al. [2]). Other (for example, probabilistic) techniques yielding a score p, $p \in [0, 1]$ may be substituted.

Core semantic matching procedure steps include:

1) 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 [6], the following describes a request for red cars manufactured between 2009 and 2011, and a particular car to be matched against this request:

$$Request \equiv Car \sqcap \exists yearMade \geq 2009 \sqcap \exists yearMade \leq 2011$$
$$Car32 \equiv FordKa \sqcap yearMade = 2010 \sqcap colour = Magenta$$

Additional knowledge about cars ($FordKa \sqsubseteq Car$) and colours ($Magenta \sqsubseteq Red$) supports the determination that the car satisfies the request. More generally, a resource description is compatible with a request if their intersection is satisfiable.

2) Quantifying Semantic Similarity: Paolucci et al. [7] and Li et al. [8] introduced 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. [5] extend the match-degree model with scoring mechanisms to support differentiation of matches falling within the same category. Descriptions of each of the match-degrees and formulas for scoring taxonomically related concepts are shown in table I.

Where two concepts do not share an ontological relationship (that is, they are disjoint), additional heuristics may be employed to measure their similarity. Proposed techniques include conflating individual comparisons of two concepts' features [9], and scoring based on the syntactic similarity of concepts' textual descriptions [10].

3) Match Algorithm: the following pseudocode describes the core steps in a semantic matching algorithm.

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

The MATCH function compares two variables X and Y, which correspond to the request and advertisement or their sub-terms respectively. A check is first made to see if X subsumes Y; if so, the algorithm terminates with a similarity score of 1. If X and Y 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 whether the concepts are ontologically or otherwise related, as per section II-A2. If not, X and Y are composites that consist of named concepts or existential restrictions. We handle these recursively by decomposing X into its constituent parts, X_i, and averaging the maximum score for each X_i, when compared with corresponding candidate components of Y, $Y_{1..m}$. Alternative implementations may consist of separating features into multiple dimensions, or considering mandatory and optional properties.

B. Supporting semantic reasoning in SAPERE

The SAPERE architecture supports pervasive services that are bound to the locality and context in which they execute by reifying data, knowledge, and events in the precise points (or region) 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 Semantic Annotations" (LSA) that continuously reflect the state of their associated components (live), which is implicitly or explicitly connected to the domain in which such information is produced, interpreted and manipulated (semantic). LSAs are reified in a networked, distributed space (called an "LSA-space") acting as the *fabric* of the ecosystem.

LSAs have a unique, system-wide identifier (LSA-id), and a content that includes the information the agent wants to manifest. They are realised as an RDF-like [11] set of triples that consist of a subject (an LSA-id), a predicate (the property name, a Uniform Resource Identifier – URI) and an object (the assigned value, a literal, URI, or locally scoped identifier). By adopting a notation resembling N3 [12], an LSA is represented, for example, as "`id p v; id q w1 w2 w3;`" where `id` is the LSA-id, property `p` is assigned to value `v`, and property `q` is assigned to values `w1`, `w2`, and `w3`.

Aspects of autonomous adaptation and management are achieved following the natural inspiration [13] 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 [14] of the kind "`P+..+P --r-->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)* constraints of a variable are lifted out to an unordered sequence of `SideConditions`, which are either "`FILTER` (exp)" or "`BIND` $(exp$ `as ?V`$)$"; *(iv)* each predicate in a triple can be prepended by either symbol +, − 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 the right-hand side patterns. Eco-laws obey a numeric transformation rate `r` representing 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 any such transformations affecting their LSAs.

A mapping of eco-laws to SPARQL/SPARUL, discussed in [15], supports their implementation in conjunction with the RDF-based encoding of LSAs. Using functionality provided by the ARQ query engine [16] and Pellet reasoner [17], which performs OWL DL reasoning using sets of core SAPERE ontologies and ontologies provided by agents acting locally.

We implement semantic matching by augmenting the ARQ query engine with an external function: `:semanticMatchDegree(?X, ?Y)`, which implements the algorithm described in section II-A3, resolving a comparison between two concepts to a match-degree value. This function is inserted directly into the eco-law language.

The complete framework is realised as a lightweight and minimal 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 are deployed in all network nodes, and apply locally [18].

III. A Framework for Realising Decentralised Resource Discovery

In this section we 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.

A. Self-organisation Patterns

Fernandez-Marquez et al. [19] provide a catalogue of self-organisation mechanism in terms of modular and reusable design patterns. This section briefly reviews those used in this paper that form the building blocks on top of which we achieve decentralised resource discovery.

1) Spreading: The Spreading pattern is for information dissemination. Spreading 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)
```

2) Aggregation: The Aggregation pattern reduces the amount of information in the system, typically to summarise data disseminated by agents over time, e.g., by spreading. It can be supported by the following eco-law:

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

3) Gradient: The Gradient pattern is a composition of the Spreading and Aggregation patterns, where information about the sender (physical or logical), 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 distance value each time, and an aggregation function retaining the less traveled LSA among competing gradient LSAs from the same source [20].

4) Chemotaxis: The Chemotaxis pattern provides a mechanism to perform motion coordination in large scale systems; it is based on the Gradient pattern: data items (or agents) are routed towards the source of a gradient via the shortest accessible path from within the gradient's extent. In our framework, movement of LSAs towards the gradient source is achieved by diffusing in the direction indicated by property `:prev` as created by Spreading eco-law above. This pattern can be the basis for more advanced strategies dynamically avoiding congested areas [20].

B. Self-organising Semantics-aware Resource Discovery

Building upon the concepts of semantic matching and self-organisation patterns, we now introduce the key steps in our approach to self-organising, semantics-aware resource discovery, which are pictorially illustrated in figure 1.

1) Establish the resource request: An application searching for a resource publishes a `request` gradient source LSA carrying information about the source R of the request: spreading and aggregation eco-laws will iteratively fire in tandem 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.

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

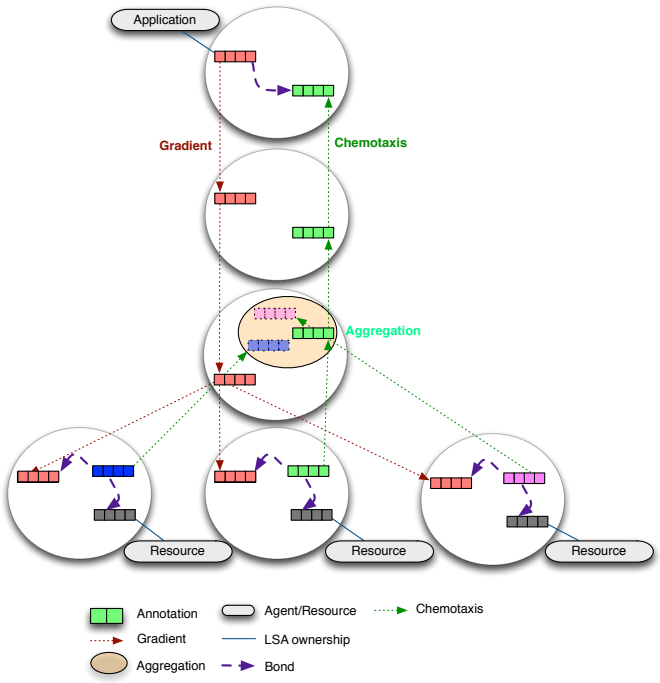

Fig. 1: Interactions in the semantic service matching: the application spreads a request gradient, at the site of each matching resource (service) a reply is produced ascending the gradient and aggregating with others.

3) Route match information to the requestor: The reply LSA is created in a way that it fires aggregation and diffusion eco-laws such that it diffuses by copying towards the request source, keeping track also of the distance travelled, namely, the chemotaxis pattern is applied to deliver a remote representation of the resource annotation matched to the request source. Note the Chemotaxis pattern is continually applied to maintain a live link between the two points in the network, request site and service site. This process ends after the request gradient expires. For example, after the request is explicitly removed by the requestor. 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.

4) Aggregate results en route by competition: In the case where a request is matched to multiple resources across a network of spaces, we wish only to return the most suitable match to the requestor at any point in time. This is achieved by applying the Aggregation pattern to any two annotations directed towards the same source, using a spatial-semantic approach, namely, considering both match-degree comparison, and distance travelled. For example, given two co-located response annotations of the form $\langle S_0, 20, 0.9\rangle$ and $\langle S_1, 5, 0.75\rangle$ (service, distance, match-degree), we might choose to retain the second which, although it has worst match-degree, is closer to the requestor.

5) Aggregate results en route by collaboration: Finally, we provide a method for augmenting standard semantic-matching with additional context – making a resource more, or less, attractive – based on factors external to the resource itself. For instance, in applications where resources represent physical point of interest (POI), we may wish to consider a small region of the space that includes many POIs more favourable than another region with a POI with highest match-degree. This is achieved by an aggregation function that "joins" two replies instead of selecting one of them. For example, given response annotations of the form $\langle S_0, 1, 0.9\rangle$ and $\langle S_1, 2, 0.75\rangle$, they can be subject to joining since they are represent services which are very near in space. Hence, we can aggregate them into a single annotation of the kind $\langle S_0 + S_1, 1.5, 0.85\rangle$, in which distance and match-degree properly summarise the original ones. An additional example strategy we do not expand on in this paper but which is an interesting subject of future work is based on the idea of decreasing the match-degree based on a calculation of how 'busy' an area of the network is (e.g., if the matched resource is to be physically navigated toward), that is, making the approach congestion-aware as in [20].

6) Receipt of replies: The requestor will eventually receive aggregated replies, providing information about (summary) distance, (summary) match-degree, service provider (or services provider), and direction to reach the service. Such information can then be used e.g., to steer the requestor agent towards a selected service, as developed in [20].

IV. Implementing the Strategy

In this section we first describe the eco-laws that support the semantic-spatial management of resource discovery, describing their behaviour and the functions they encapsulate, and present a preliminary simulation.

A. Eco-laws to support match-degree

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 [20]. We instead focus on the three eco-laws that incorporate the key strategies described in previous section, namely, the generation of replies along with a match-degree, and the competition-/collaboration-based aggregation of them as they are routed towards the requestor. They are presented in Figure 2, and described in turn. Note such eco-laws make use of a default name-space, omitted in the corresponding URIs, corresponding to the ontology of resource discovery, and the `sos` namespace incorporating all the concepts relating to self-organisation patterns.

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 is of type `:reply`, it points (via so-called bonds) to the request and service LSAs, it is also of type `sos:ascend` (with initial distance 0 so as to ascend the gradient to which `REQ` is part. It also describes the match-degree `?MD`, computed by external 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

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 + ?SER :type :service; :content ?S
--?F-->
?REQ + ?SER +
?REP :type :reply; :service ?SER; :request ?REQ; :match_degree ?MD;
     sos:type sos:ascend; sos:direction ?REQ; sos:dist "0"
BIND(:semanticMatchDegree(?R, ?S) AS ?MD)
FILTER(?MD > 0)
BIND(:replyRate() AS ?F)

% [AGGREGATE-BOOST] Joins information from two replies belonging to the same service cluser
?A1 :type :reply; :request ?R; :service ?S1; sos:dist ?D1; :match_degree ?MD1 +
?A2 :type :reply; :request ?R; :service ?S2; sos:dist ?D2; :match_degree ?MD2 +
-->
?A1 :service ?S; sos:dist = ?D; :match_degree = ?MD
FILTER(:near(?D1,?D2)) .                    % The two services are near
BIND (:center(?D1,?MD1,?D2,?MD2) AS ?D) .   % Let D be the aggregated distance
BIND (:tconorm(?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; :request ?R; :service ?S1; sos:dist ?D1; :match_degree ?MD1 +
?A2 :type :reply; :request ?R; :service ?S2; sos:dist ?D2; :match_degree ?MD2 +
-->
?A :service ?S1; sos:dist ?D1; :match_degree ?MD1
FILTER (!:neighbour(?D1,?D2)) .             % The two services are not neighbours
FILTER (:stronger(?D1,?MD1,?D2,?MD2)) .     % Service A1 has a stronger match-distance pair
```

Fig. 2: Eco-laws for handling request-reply match, and for distributed aggregation.

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 [AGGREGATE-BOOST] joins two reply LSAs whenever they are 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 distance values are such that they are perceived as "near" by external function `:near`, 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`, distance `?D` is computed by the `:center` function, and finally match-degree is computed by the `:tconorm` function. As a reference implementation for such functions (used in our simulations, and to be compared with other approaches in the next activities of this research) we consider: `:near` simply check that the two distances are both below a given threshold; `:join` appends the two arguments yielding the list of all service identifiers; `:center` performs a weighted mean of `?D1` and `?D2` based on match-degrees, namely, $(?D1 * ?MD1 + ?D2 * ?MD2)/(?M1 + ?M2)$; and finally `:tconorm` implements a t-conorm (a function to perform "union" to fuzzy results [21]), and in particular the so-called product logic t-conorm $?MD1 + ?MD2 - ?MD1 * ?MD2$.

Eco-law [AGGREGATE-CHOOSE] joins two reply LSAs whenever they are *not* perceived as belonging to a unique cluster of services. Similarly to the previous law, it takes two reply LSAs for the same requestor `?R`, whose distance values are such that they are not perceived as "near" by external function `:near`, and selects the one with the stronger semantic-spatial situation (discarding the other). This is computed by external function `:stronger`, which in our implementation checks whether $(?MD1 * k/(?D1 + k)) \geq (?MD2 * k/(?D2 + k))$ for some fixed parameter k. This compares the two match-degrees, but penalises the one having greater distance (e.g., when $D = k$ the actual match-degree is halved).

B. Simulated Illustration

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.

The simulated scenario aims to provide early validation of the idea presented above, investigating model and functions (matching and aggregator operators) proposed in section IV-A for selecting the best services among services matching a request—considering both the match-degree and the physical distance between the requestor and services. In particular we simulated a scenario where: *(1)* a requestor is placed in the centre of a 20×10 grid. *(2)* services are spatially distributed over the grid such that one service is isolated while a set of others form a cluster in an opposite region of the grid, with respect to the source. Each node of the grid simulates the behaviour of an LSA-space, containing in particular eco-laws for *(i)* gradient definition, *(ii)* matching and *(ii)* reply propagation and aggregation, implemented as described in section IV-A. Services in the cluster provide replies for the request with a match-degree randomly chosen in the range $[0.2, 0.6]$, while the isolated service matches the request with $MD = 0.95$. Given this scenario we simulated different settings, placing the isolated service at different distances from the source while leaving the cluster position fixed.

For model parameters, we defined a service cluster, as used in the `[AGGREGATE-BOOST]` eco-law, as those services physically closer than $1/3$ of the highest gradient value, i.e.,

the maximum distance to the request ($Dmax$). The other parameters $k, D1, D2$ are varied to analyse system behaviour.

Figure 3 shows how varying the value for the k parameter of the `:stronger` predicate modifies the system's preference for the cluster and single service choices. k strongly affects the impact distance has upon to the resource match-degree: the higher the value of k, the lower the influence of distance in the selection process.

This simulation serves only as an illustration of the potential of the approach presented. Future work will aim to simulate factors affecting implementation in large pervasive systems, such as: *(i)* how requestor, service, and node mobility affects behaviour, *(iii)* how system properties such as discovery time and bandwidth are affected by the matching and aggregation operators employed, *(ii)* how node density and connectivity impacts performance, and *(iv)* how alternative and multiple contextual modifiers (such as crowdedness) may be incorporated modularly.

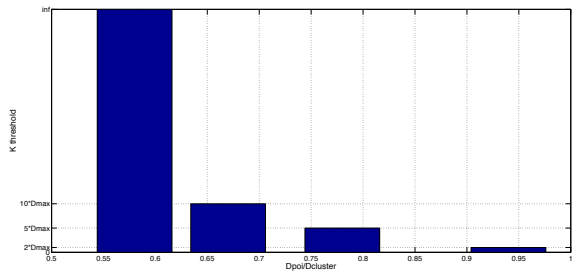

Fig. 3: The effect of k values on single/cluster selection.

Simulations evaluating fault tolerance, and evaluating performance in a mobile setting remain as future work.

V. Conclusion

Using bio-inspired self-organising patterns as a communication mechanism, this paper proposed a novel approach to fully decentralised semantics-aware resource discovery in a mobile setting. Information about the match-degree between requests and resources directly affects the application of the patterns and additional context, namely physical distance, is integrated to provide on-the-fly, distance-aware resource selection. A simulated proof of concept shows how the impact of distance on the match-degree can be controlled.

In the future we aim to build upon this work, generalising our approach to support the augmentation of match-degrees with arbitrary forms of context, not only distance. This potentially leads to the construction of a library of such augmenter functions, where an application may select from among a set of possible concerns that may affecting resource selection.

Acknowledgment

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

References

[1] Franco Zambonelli et al. Self-aware pervasive service ecosystems. *Procedia Computer Science*, 7:197–199, December 2011.

[2] Dimitrios Skoutas, Alkis Simitsis, and Timos K. Sellis. A ranking mechanism for semantic web service discovery. In *IEEE SCW*, pages 41–48, 2007.

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

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

[5] Ayomi Bandara, Terry Payne, David De Roure, Nicholas Gibbins, and Tim Lewis. Semantic resource matching for pervasive environments: The approach and its evaluation. 2008.

[6] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[7] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia 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, UK, 2002. Springer.

[8] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 331–339, New York, NY, USA, 2003. ACM.

[9] Angela Schwering. Hybrid model for semantic similarity measurement. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3761 of *Lecture Notes in Computer Science*, pages 1449–1465. Springer Berlin Heidelberg, 2005.

[10] Matthias Klusch and Frank Kaufer. Wsmo-mx: A hybrid semantic web service matchmaker. *Web Intelligence and Agent Systems*, 7(1):23–42, January 2009.

[11] Eric Miller and Frank Manola. RDF primer. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[12] Tim Berners-Lee and Dan Connolly. Notation3 (N3): A readable RDF syntax. W3C team submission, W3C, March 2011. http://www.w3.org/TeamSubmission/n3/.

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

[14] Jean-Pierre Banâtre and Thierry Priol. Chemical programming of future service-oriented architectures. *JSW*, 4(7):738–746, 2009.

[15] Mirko Viroli, Franco Zambonelli, Graeme Stevenson, and Simon Dobson. From SOA to pervasive service ecosystems: an approach based on semantic web technologies. In *Adaptive Web Services for Modular and Reusable Software Development: Tactics and Solution*. 2012. To Appear.

[16] ARQ - a SPARQL processor for Jena. http://jena.sourceforge.net/ARQ/.

[17] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5:51–53, June 2007.

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

[19] Jose Luis Fernandez-Marquez, Giovanna Di Marzo Serugendo, Sara Montagna, Mirko Viroli, and Josep Lluis Arcos. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing*, pages 1–25, 2012.

[20] Mirko Viroli, Danilo Pianini, Sara Montagna, and Graeme Stevenson. Pervasive ecosystems: a coordination model based on semantic chemistry. In Sascha Ossowski, Paola Lecca, Chih-Cheng Hung, and Jiman Hong, editors, *Proceedings of SAC 2012*, Riva del Garda, TN, Italy, 26-30 March 2012. ACM.

[21] Fernando Bobillo and Umberto Straccia. Reasoning with the finitely many-valued ukasiewicz fuzzy description logic. *Information Sciences*, 181(4):758 – 778, 2011.

[22] Danilo Pianini, Sara Montagna, and Mirko Viroli. A chemical inspired simulation framework for pervasive services ecosystems. In Maria Ganzha, Leszek Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 675–682, Szczecin, Poland, 18-21 September 2011. IEEE Computer Society Press.