Dynamic Adaptive Middleware Services for Service Selection in Mobile Ad-Hoc Networks

Rogério Garcia Dutra and Moacyr Martucci Jr.

Department of Computer and Digital Systems Engineering (PCS),
Escola Politécnica, Universidade de São Paulo,
Av. Prof. Luciano Gualberto, trav 3, no 158, São Paulo, SP, Brasil
rogdutra@gmail.com, moacyr.martucci@poli.usp.br

Abstract. Dynamic adaptive service selection became a key necessity for most mobile middlewares based on functional services properties. Well-grounded algorithms for datamining were used for unsupervised selection of services clusters with similar non functional properties, adaptive induction of decision trees for supervised selection of quality of service (QoS) parameters relationship and adaptive fuzzy inference to manage uncertainty in QoS measures. These algorithms, encapsulated as services, compose a middleware solution for mobile ad-hoc networks service selection, using Service-Oriented Architecture (SOA) approach.

Keywords: Adaptation and Service Selection; Mobile Ad-Hoc Network QoS Awareness; Datamining Algorithms for Unsupervised and Supervised Learning; Fuzzy Inference; Service- Oriented Architecture (SOA) Middleware.

1 Introduction

Well-grounded datamining algorithms are ready-made tools, that have been peer-reviewed, widely used and tested to tackle data heterogeneity and sparseness, supporting the discovery process of knowledge-based interactions and relationships from high volume databases.

Pervasive computing using mobile devices faces similar challenges, where heterogeneity is characterized by lack of service and communication standards. A *service* is any tangible or intangible facility a device provides that can be useful for any other device. Services comprise those for software and hardware resources, which move at high or low speeds or even remain stationary, entering and leaving the system when switched on or off in the network. Sparseness and uncertainty capabilities for service discovery, are key challenges in the mutable nature of mobile ad-hoc networks (MANETs).

The nodes of MANETs intercommunicate through single-hop and multihop paths in a peer-to-peer fashion. Intermediate nodes between a pair of communicating nodes (providers and consumers) act as routers. The nodes are mobile, so the creation of routing paths is affected by the addition and deletion of nodes. The topology of the network may change rapidly and unexpectedly, once no fixed infra-structure is used.

Y. Cai et al. (Eds.): Mobilware 2010, LNICST 48, pp. 189–202, 2010.

In MANETs environment, service discovery would enable devices and services to properly discover, configure, and communicate with each other. Discovery comprises search and selection. These two mechanisms can be independent or integrated. For example, a consumer might first search for all instances of a service provider and then select a suitable service, or might perform the two functions simultaneously. Although service selection is a basic feature for service discovery approaches, it has been underestimated or simply ignored in most of discovery solutions found in literature.

Usually, a consumer issues a query to search services based on functional properties, advertised by service providers or intermediate nodes in the network, resulting in a set of similar services. To complete the discovery process, a selection based on additional service non functional properties is necessary. If this selection is not performed properly, the search will generate non optimized results, causing an unnecessary overhead in MANETs environment or low Quality of Service (QoS) perception from the consumer point of view.

To overcome these challenges, this paper propose a novel selection solution called Dynamic Adaptive Middleware Services for Service Selection (DAMS-SS) in MANETs, to satisfy the following requirements:

- Cluster search results, based on unsupervised learning of Self-Organizing Map algorithm, without consumer interaction or hard-coded assumptions;
- Define hierarchical cluster relationships, using adaptive and incremental supervised learning of an Adaptive Decision Tree algorithm;
- Adapt consumer service request, managing uncertainty in QoS metrics definitions from the consumer perspective, using a Fuzzy Inference algorithm.

The expected benefits from DAMS-SS solution are:

- Enhance service selection capabilities of existing functional middleware solutions, encapsulating datamining algorithms as middleware services based on a service architecture;
- Transform data gathered from MANETS into comprehensible information to support consumer decision on best service choice selection;
- Propose a structured process for service search refinement combined with a reactive and proactive selection method.

The rest of this paper is organized as follows. Section 2 describes the related work and current research about service selection and algorithms used for service mining. Section 3 describes the proposed middleware architecture and the service selection process. Section 4 presents the results and section 5 the conclusion and future work.

2 Related Work for Service Selection and Mining Algorithms

Although service selection is a basic feature for service discovery approaches, it has been many times underestimated or simply ignored. In a service discovery survey [1] for MANETs, 12 works were evaluated, but only 3 proposed selection algorithms, although service search and selection are often integrated. In [2] demonstrated that proper integration improves overall network performance by localizing network communication, thus reducing interference and allowing multiple concurrent transmissions in different parts of the network.

However, none of these works proposed any kind of intergration of service mining techniques, to enhance service selection. In [3], a middleware that exploits machine learning techniques to learn how to perform "on the fly" translations across ontologies, removing the unrealistic assumption that devices exchange knowledge by means of a shared ontology (or a set of statically known ones). It uses Kohonen's *Self-Organizing Map* (SOM) [4] for service cluster exploration, due to its unsupervised learning capability. The drawback of SOM is the inability to manage uncertainty in data and explain the relations of gathered clusters, due to opaque nature of its algorithm, as described in section 2.1.

To manage uncertainty, usually generated by non clearly defined service consumer requests, [5] proposed a fuzzy service adaptation engine for context-aware mobile computing middleware biased on known set of rules or policies extracted from historical data. The dependence from historical data generates a hard issue, due to its unpredictable nature of MANETs. To overcome this issue, an *Adaptive Network-based Fuzzy Inference Systems* (ANFIS) [6], combining supervised neural networks and fuzzy logic, is necessary to generate a set of rules from the SOM cluster results, to avoid a historical data analysis, as described in section 2.3.

This set of rules, based on all rules combinations, can potentially generate a huge amount of possibilities, causing a overhead in the fuzzy inference engine and impacting in ANFIS performance. To mitigate this pitfall, the rules must be induced and evaluated incrementally based on SOM cluster relationships. An *Adaptive Decision Tree* algorithm [7] can perform this task, as described in the section 2.2.

2.1 Self-Organizing Map (SOM) Algorithm

Due to its unsupervised learning capabilities, the Self-Organizing Map (SOM) is usual tool chosen for exploratory phase of datamining [8]. It projects input space on prototypes of a low-dimensional regular grid that can be effectively used to visualize and explore properties of the data. When the number of SOM units is large, to facilitate quantitative analysis of the map and the data, similar units need to be grouped, i.e., clustered. A two-stage procedure—first using SOM to produce the prototypes that are then clustered in the second stage—is found to perform well when compared with direct clustering of the data and to reduce the computation time.

First, a large set of prototypes—much larger than the expected number of clusters—is formed using SOM or some vector quantization algorithm. The prototypes can be interpreted as "protoclusters", which are in the next step combined to form the actual clusters. Each data vector of the original data set belongs to the same cluster as its nearest prototype.

SOM consists of a regular, usually two-dimensional (2-D), grid of map units. Each unit i is represented by a prototype vector $m_i = [m_{i1},\ldots,m_{id}]$, where d is the input vector dimension. The units are connected to adjacent ones by a neighborhood relation. The number of map units, which typically varies from a few dozen up to several thousand, determines the accuracy and generalization capability of SOM. During training, the SOM forms an elastic net that folds onto the "cloud" formed by the input data. Data points lying near each other in the input space are mapped onto nearby map units. Thus, SOM can be interpreted as a topology preserving mapping from input space onto the 2-D grid of map units.

SOM could be trained iteratively, where at each training step, a sample vector is randomly chosen from the input data set. Distances between and all the prototype vectors are computed. The bestmatching unit (BMU), is the map unit with prototype closest to the sample vector, using a distance measure, usually Euclidian distance. The update rules and training parameters for vector prototypes can be found in [8], since clustering SOM units after SOM training is better than clustering services directly.

The primary benefit of the two-level approach is the reduction of the computational cost. Another benefit is noise reduction. The prototypes are local averages of the data and, therefore, less sensitive to random variations than the original data, although it cannot avoid uncertainty in data.

The two main ways to cluster data, make the partitioning are hierarchical and partitive approaches. The hierarchical methods can be further divided to agglomerative and divisive algorithms, corresponding to bottom-up and top-down strategies, to build a hierarchical clustering tree. Partitive clustering algorithms divide a data set into a number of clusters, typically by trying to minimize some criterion or error function. The number of clusters is usually predefined, but it can also be part of the error function [9].

Partitive methods are better than hierarchical ones in the sense that they do not depend on previously found clusters. On the other hand, partitive methods make implicit assumptions on the form of clusters. To select the best one among different partitionings, each of these can be evaluated using some kind of validity index.

Several indices have been proposed [10], [11]. The Davies–Bouldin index [12] was used, because is suitable for evaluation of k-means [9] partitioning because it gives low values, indicating good clustering results for spherical clusters. If desired, some vector quantization algorithm, e.g., k-means, can be used instead of SOM in creating the first abstraction level. Other possibilities include the following. Minimum spanning tree SOM [13], neural gas [14], growing cell structures [15], and competing SOM's [16] are examples of algorithms where the neighborhood relations are much more flexible and/or the low-dimensional output grid has been discarded.

In any case, the pruning produces a more clear dendrogram but still does not provide a unique partitioning. This is not a problem if the objective is only to find a set of interesting groups. If necessary, a final partitioning can be selected kind of interactive tool, such as a decision tree, since SOM opaque algorithm cannot incrementally define the relationship of gathered clusters.

2.2 Adaptive Decision Tree (ADAPTREE) Algorithm

According [7], decision trees are widely used as a knowledge representation tool in machine learning, mainly for their clean graphical representation, which captures, some aspects of the human decision process. Algorithms for inducing decision trees from examples, a major problem in machine learning, include ID3 and C4.5 [17], CART [18] and ITI [19], the last one being an incremental inducer.

ADAPTREE is a decision tree induction algorithm based on adaptive finite state automata, extended with some non-syntactical characteristics to handle continuous features and statistical generalization. ADAPTREE can be incrementally trained to solve classification problems. The general idea is to handle each training and testing instance as a string and consider the decision tree itself as a special kind of adaptive finite state automaton, with the initial state of this automaton corresponding to the root of a classical decision tree.

The core of the ADAPTREE learning strategy, including the generalization mechanism based on conditional probabilities estimates, which are calculated dynamically, is incremental. Hence, the training and testing instances could be presented one by one, interchangeably. In fact, the strategy could also be viewed as a different kind of instance-based learning [20], with some resemblance to the approaches based on k-d trees [20]. However, if the attributes are to be reordered, the automaton should suffer some major modifications from time to time.

Datasets containing continuous value attributes should also present a problem to the incrementality of ADAPTREE, at they must be previously discredited. In the current implementation, a discretization method, described in [19], is automatically performed on each continuous feature present in the dataset, before the learning takes place. This method employs a supervised (use class information) approach based on recursive entropy minimization and a minimum description language (MDL) stopping criteria [21], and so, depends on the existence of some training examples (supervised learning).

ID3 algorithm was originally used as tree induction algorithm by ADAPTREE, but according [17], if an attribute has a large number of possible values, the higher will be the information gain. To avoid this kind of distortion, C4.5 algorithm uses the ratio of entropy information gain, generating better classification results compared to ID3. To capture this benefit, the original ADAPTREE was modified to use the ratio of entropy information gain, but without pruning the branches as in C4.5 algorithm, resulting in small trees due to the incremental strategy.

Due to mutable nature of MANETs, the training set is seldom not known, requiring an unsupervised learning technique as SOM algorithm, to define the number of clusters or classes and map each service to only one class. This kind of map is called crisp clustering, where each data sample belongs to exactly one cluster. Fuzzy clustering [22] is a generalization of crisp clustering where each sample has a varying degree of membership in all clusters, as described in the following section.

2.3 Adaptive Network-Based Fuzzy Inference Systems (ANFIS) Algorithm

According [6], fuzzy if-then rules or fuzzy conditional statements are expressions of the form IF A THEN B, where A and B are labels of fuzzy sets [23] characterized by appropriate membership functions. Due to their concise form, fuzzy if-then rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision.An example that describes a simple fact is “If pressure is high, then volume is small”, where pressure and volume are linguistic variables [24], high and small are linguistic values or labels that are characterized by membership functions.

Another form of fuzzy if-then rule, proposed by Takagi and Sugeno [25], has fuzzy sets involved only in the premise part. Where, again, high in the premise part is a linguistic label characterized by an appropriate membership function. However, the consequent part is described by a nonfuzzy equation of the input variable, velocity.

Both types of fuzzy if-then rules have been used extensively in both modeling and control. Through the use of linguistic labels and membership functions, a fuzzy if-then rule can easily capture the spirit of a “rule of thumb” used by humans. Basically a fuzzy inference system is composed of five functional blocks:

- A rule base containing a number of fuzzy if-then rules;
- A database which defines the membership functions of the fuzzy sets used in the fuzzy rules;

- A decision-making unit which performs the inference operations on the rules;
- A fuzzyfication interface which transforms the crisp inputs into degrees of match with linguistic values;
- A defuzzification interface which transform the fuzzy results of the inference into a crisp output.

Several types of fuzzy reasoning [26] have been proposed in the literature, but Takagi and Sugeno's (TS) fuzzy inference system (FIS) is used, once the output of each rule is a linear combination input variables plus a constant term, and the final output is the weighted average of each rule's output. Due to its simplicity and high performance, TS FIS is suitable for ANFIS building.

The acronym ANFIS derives its name from adaptive neuro-fuzzy inference system. Using a given input/output data set, the toolbox function ANFIS constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back propagation algorithm alone, or in combination with a least squares type of method. This allows your fuzzy systems to learn from the data they are modeling.

A network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map.

The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure (usually defined by the sum of the squared difference between actual and desired outputs). ANFIS uses either back propagation or a combination of least squares estimation and backpropagation for membership function parameter estimation.

In the proposed architecture for service selection, ANFIS uses the If-Then rules derived from ADAPTREE, where the service clusters were resulted from SOM unsupervised learning algorithm.

3 DAMS-SS Architecture and Selection Process

Search and selection tasks are normally performed at a intermediate layers between the network and application layer, named middleware. To reduce middleware complexity, service architectures were proposed to establish standards for service provisioning and consumption.

Service-Oriented Architecture (SOA) is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable standard interfaces [27]. In a most known implementation of a Service-Oriented Architecture, a service description is based on the a standard called *Web Services Definition Language* (WSDL) to

enable service discovery based on functional attributes in internet. Since WSDL supports only syntactical discovery, other languages were proposed to support semantic discovery based on functional ontologies, where *Ontology Web Language for Services* (OWL-S) [27] is one of the most known languages used for this purpose.

Additionally to service language definitions, the Quality of Service (QoS) description publishes important functional and non-functional service quality attributes. WSDL did not support non-functional attributes and OWL-S ontologies can be extend to support QoS features. Other descriptions languages were created to support both attributes, such as *Web Service Offering Language* (WSOL) [28].

From the service consumer perspective [29], the following four generic QoS criteria are considered for non terminal nodes in MANETs:

- **Availability** – The availability of a service is the probability that service is usable.
- **Price** – The price of a service is the fee that a service requester has to pay for using the service. The value of this QoS parameter is given by the service provider.
- **Reliability** - The reliability of a service is the probability that a service request is correctly responded, namely, the requester has received the expected results, within the maximum expected time frame indicated in the service description.
- **Execution Delay** - The delay of a service is a measure of duration between the time point when a service request is sent out and the time point when the results are received by the requester.

According [29], these criteria could be applied for atomic service or composed service selection. From the network nodes perspective, the following secondary criteria could be considered as a search refinement:

- **Node availability -** The availability of a node to its neighbor nodes is highly related to the node's mobility—here it is assumed that battery power is not a concern and as such a node will not out of reach, due to battery exhaustion.
- **Network Delay -** The delay of a packet in a network is the time it takes the packet to reach the destination after it leaves the source. Similar to that in fixed networks, the delay in wireless mobile networks is the sum of time spent at each relay on the route.
- **Network reliability –** The reliability is measured in this paper by the packet loss rate of the wireless link via which the service host is connected to the outside world.

Publication of such information about available services provides the necessary means for discovery, selection, binding, and composition of services. A discovery agency or service registry stores published descriptions, where *Universal Description Discovery & Integration* (UDDI) [30] is the registry standard in SOA for Web Services. A centralized UDDI discovery agency model implies some challenges to support dynamic discovery in MANETs, once it does not define any mechanism to propagate discovery queries toward the site where the corresponding information is stored. The efficacy of queries depends on the completeness of the information stored on the registry that is being contacted, on the precision of the query, and also on the ability of the requester to explicitly query different registries.

In order to implement a novel QoS based selection in MANETs, the proposed architecture extends SOA concepts, wrapping the combination of mining algorithms as middleware services to support and enhance exiting functional search solutions, as described in the following.

Dynamic Adaptive Middleware Services for Service Selection (DAMS-SS) architecture and process are described in the picture 1.

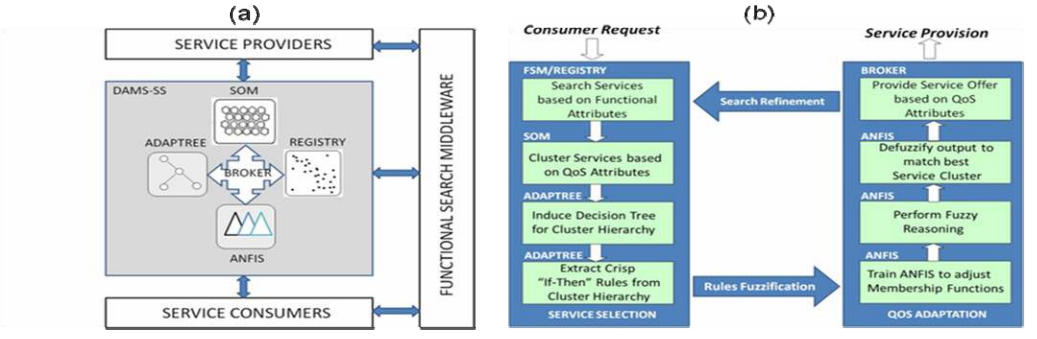

Fig. 1. DAMS-SS Architecture (a) and Selection Process (b)

The main components in fig 1(a) are SOM, ADAPTREE and ANFIS algorithms encapsulated as services, using a service description language. These services can be assessed by service providers and consumers directly through the BROKER service, to perform the selection. Additionally, a service REGISTRY was necessary to store temporary the services resulted from a functional query from consumers to its Functional Search Middlewares (FSM). Many FSM solutions can be found in the literature, as SAMOA [31], MOBISOC [32], CAMPE [33], for example.

Based on SOA guidelines, the Web service description used is WSDL and BROKER must allow distributed, content-based, publish and subscribe messaging for query and advertise in MANETs dynamic environments.

There are many BROKER proposals in the literature, but REDS [34] fulfills all the requirements above. REDS, differently from the other similar middleware, natively offers the possibility of replying to messages. This feature can be very useful to transfer the result of a query from the resource owner to the requester, thus completely addressing the requirements of query–advertise.

Another key feature of REDS is the internal structure of its brokers, which are organized as a set of modules that implement well-defined interfaces and encapsulate the major aspects of Content-Based Routing (CBR). CBR holds the promise of addressing, at the same time and with a single routing infrastructure, the two issues we identified in SOAs: providing support to an asynchronous, publish–subscribe interaction style, and enabling complex searches to be executed in a completely distributed environment.

Fully content-based WS-Notification subscriptions and messages can be easily managed by *RedsNotificationBroker* (i.e., routed by REDS), while precise queries for exactly the services required can be formulated via *RedsUDDINode*, which routes them toward the right service providers. *RedsUDDINode* module was used also to implement service REGISTRY, as a distributed UDDI searched services.

As in datamining, the selection is a cyclic process, which starts with a consumer issuing a request and ends with a service offer to match it, as describe in Fig. 1(b).

In a reactive selection mode, a service consumer issues a service request based on functional attributes. The FSM reacts, searching in REGISTRY first, than in MANET nodes, services whose functional attributes best match the consumer request. The select services are stored or updated in the service REGISTRY.

In a proactive selection mode, FSM search available services within a search scope: context-based (e.g. SAMOA), people-based (e.g. CAMPE) or a combination of both (e.g. MOBISOC), instead reacting to a service request.

SOM service reads the REGISTRY, using BROKER services, and cluster available services using its unsupervised learning algorithm. The clusters are used to induce an adaptive decision tree using ADAPTREE service, generating a service cluster hierarchy, composed by QoS attributes on non terminal nodes, and service classes on terminal nodes (leafs).

The search refinement will be necessary when MANET became very instable, and only QoS parameters perceived by the user are not enough to fulfill consumer demands. SOM can cluster services based on both types of QoS criteria, if necessary to speed-up the search process, or use first user parameters than network parameters in an iterative mode.

Reading top-down the decision tree, If-Then rules are created by ADAPTREE and fuzzyfied to train ANFIS based on Membership Functions (MF) adjustments. Once adjusted, the MFs will allow ANFIS to match the best service cluster to consumer request fuzzy QoS statements as follows:

From users perspective:

- The highest availability and reliability;
- The lowest price and execution delay.

From Network perspective:

- The highest node availability and network reliability;
- The lowest network delay.

The ANFIS output defuzzification based on QoS parameters will set the boundaries of a "*n*" dimensional (N-D) fuzzy decision surface or region, where *n* is equal to the number of input parameters.

4 Implementation and Results

To implement a prototype of SOM and ANFIS services, Matlab® release 2006a SOM Toolbox 2.0 [36] and FUZZY Toolbox [6] was used. Matlab® was used also to simulate a MANET environment [37]. ADAPTREE was implemented using WEKA function libraries [38] and free software available from the author [7].To test core service selection algorithms, we assume that the functional search was done by some FSM mentioned earlier, and BROKER already stored the searched services in REGISTRY. SOM service evaluates 4 QoS parameters from user´s perspective of a sample 469 services [29] database, and identify 4 clusters, based on Davies-Bouldin index minimum value, as illustrated in Fig. 2(a).

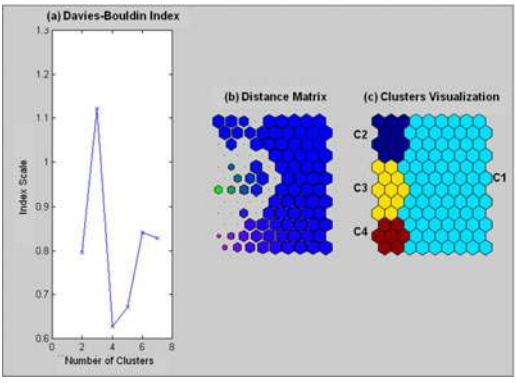

Fig. 2. SOM service results: (a) Davies-Bouldin index, (b) Distance Matrix and (c) Cluster Visualization

Fig. 2(b) illustrates the distance matrix between SOM units for cluster distance definition [8] and (c) the 4 clusters (C1 to C4) visualization over SOM units matrix. SOM unsupervised training algorithm identifies a Cluster for each service, used as input for ADAPTREE supervised training, as illustrated in Fig. 3.

In Fig. 3, "Ci:*X*" represents the number of services in each Cluster "i". "Low" and "High" represents the fuzzyfication rule of each induced branch of Decision Tree. *Info(X,T)* represents the information entropy measure used by ADAPTREE to induce another tree node, where "*X*" represents the number of services in each tree node and "*T*" the total amount of searched services in REGISTRY.

Reading the Decision Tree from root, then top-down scan until the leafs, the fuzzy "If-Then" can be collected as described in figure 4(a). The tree graphical representation

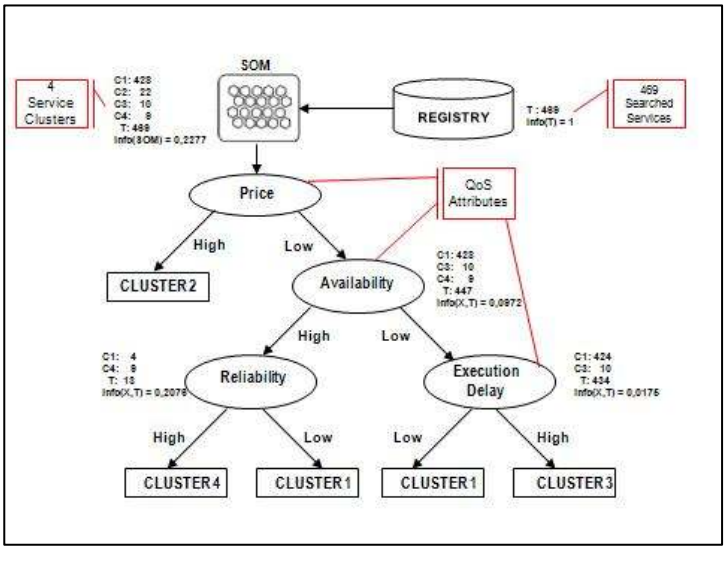

Fig. 3. Induced Decision Tree from SOM Clusters

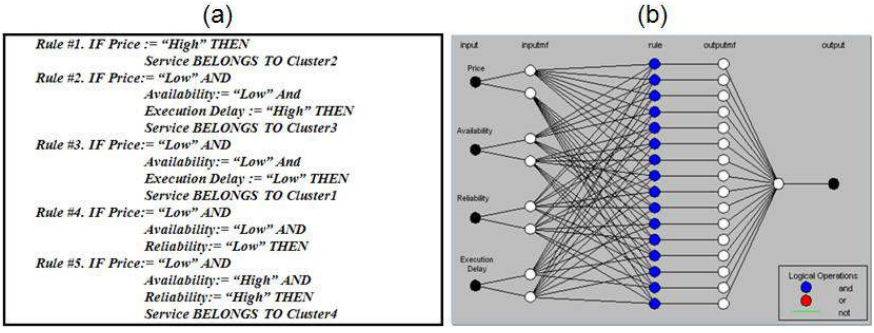

Fig. 4. Fuzzy "If-Then" rules from induced Decision Tree generated from ADAPTREE (a) and (b) structure of ANFIS system induced by ADAPTREE

is not necessary to generate the rules described in figure 4(a), but helps to understand ADAPTREE rule induction mechanism based on entropy gain.

The 5 rules described in figure 4(a), are a subset of 16 possible rules combining the 4 parameters with possible value "High" and "Low. These rules were used as inputs to train ANFIS output membership functions (MF), as illustrated in fig. 4(b).The "AND" logical operator was used to compute the "min" function of the 4 inputs, due to the inductive nature of the decision tree. The 469 services set in REGISTRY was divided into 3 parts randomly: 200 services for ANFIS training;169 services for ANFIS testing and 100 services for ANFIS checking.

The output results from each set of training can be found in Fig. 5, where "Output numbers" represent the identification of each service cluster (from 1 to 4).

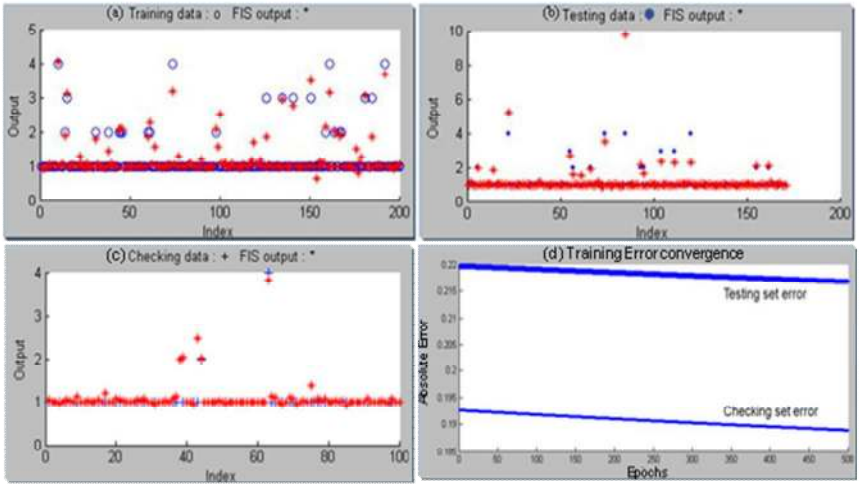

Fig. 5. (a) Output results from training and (b) Testing data set (c) Output results from checking data set and (d) Training Error convergence

In fig. 5(a), the output of ANFIS is compared to its related service cluster, as defined by SOM service. In fig. 5(b), another set of services is used for testing ANFIS after MF training. In fig. 5(c), another set of services, different for the training set and testing set, is used for checking ANFIS output accuracy. The testing and checking error decrease is slow, as illustrated in fig. 5(d), not justifying a increase in the number of training epochs.

Once ANFIS supervised training is finished, the output defuzzification, based on membership functions, set the boundaries of a fuzzy decision surface, as illustrated in fig. 6. In the fig. 6(a), Price and Availability QoS parameters were combined to define a 3-D fuzzy decision surface. In the fig. 6(b), Price and Execution Delay were combined to define another 3-D fuzzy decision surface. The combination of all 4 QoS input parameters and output, set the boundaries of a 5-D fuzzy decision region, where consumer request will be adapted to select the best service cluster based on consumer demands.

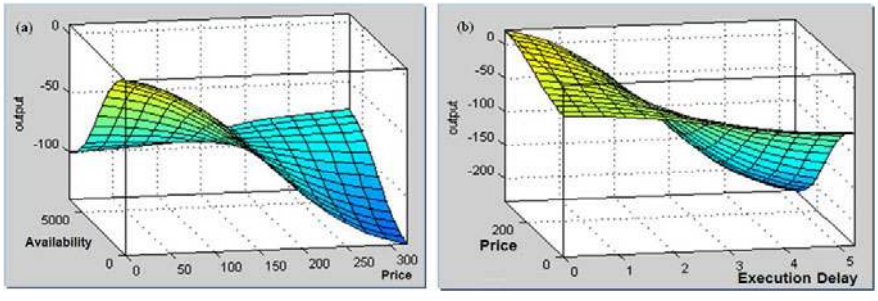

Fig. 6. Fuzzy decision surface combining different service QoS parameters

5 Conclusion and Future Work

Dynamic Adaptive Middleware Services for Service Selection (DAMS-SS) is a novel proposal to support service discovery in MANETs, based on a combination of tree well-grounded dataming algorithms. A Proof of Concept (PoC) implementation demonstrated DAMS-SS feasibility to enhance service selection process, combining unsupervised and supervised algorithms to perform fuzzy clustering based on incremental rule induction.

The proposed reactive and proactive selection method, supported by a service oriented architecture solution, where datamining algorithms were encapsulated as middleware services, refining current functional search middleware solutions.

The combination of SOM, ADAPTREE and ANFIS transformed data, gathered from MANETs, into comprehensible information to support consumer decision on best service choice selection, while reducing the drawbacks of standalone service mining implementations, as describe in the literature. Comparisons to other approaches were not possible, due to the novelty and uniqueness nature of proposed service mining algorithms combination.

Our future work includes a implementation on real mobile devices to evaluate algorithms memory consumption and possible performance issues. To measure the

trade-off between accuracy and usability, we intend to investigate and experiment with more QoS parameters, for example, networks parameters, evaluating the advantages and disadvantages of proposed iterative selection process for MANETs environments.

Although designed for MANETs, the proposed solution could also be used for service selection in distributed environments with fixed infra-structure networks, to support discovery in other service-oriented architectures, such as cloud computing, once the combination of mining algorithms could be encapsulated using any service language description.

References

1. Mian, A.N., et al.: A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks
2. Varshavsky, A., et al.: A Cross Layer approach to Service Discovery and Selection in Manets. In: Proc. 2nd Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS 2005). IEEE Press, Los Alamitos (2005)
3. Capra, L.: MaLM: Machine Learning Middleware to Tackle Ontology Heterogeneity. University College London (2005)
4. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1995)
5. Cheung, R., et al.: A fuzzy service adaptation engine for context-aware mobile computing middleware. International Journal of Pervasive Computing and Communications 4(2), 147–165 (2008)
6. Jang, J.S.R.: Adaptive Network-base Fuzzy Inference System. IEEE Transactions on Systems, Man, and Cybernetics 23(3) (May/June 1993)
7. Pistori, H., Neto, J.J., Pereira, M.C.: Adaptive Non-Deterministic Decision Trees: General Formulation and Case Study. INFOCOMP Journal of Computer Science, Lavras, MG (2006)
8. Vesanto, J., Alhoniemi, E.: Clustering of the Self–Organizing Map. IEEE Transactions on Neural Networks 11(3), 586–600 (2000)
9. Buhmann, J., Kühnel, H.: Complexity optimized data clustering by competitive neural networks. Neural Comput. 5(3), 75–88 (1993)
10. Bezdek, J.C.: Some new indexes of cluster validity. IEEE Trans. Syst., Man, Cybern. B 28, 301–315 (1998)
11. Milligan, G.W., Cooper, M.C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika 50(2), 159–179 (1985)
12. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Patt. Anal. Machine Intell. PAMI-1, 224–227 (1979)
13. Kangas, J.A., Kohonen, T.K., Laaksonen, J.T.: Variants of self-organizing maps. IEEE Trans. Neural Networks 1, 93–99 (1990)
14. Martinez, T., Schulten, K.: A neural-gas network learns topologies. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Artificial Neural Networks, pp. 397–402. Elsevier, Amsterdam (1991)
15. Fritzke, B.: Let it grow—Self-organizing feature maps with problem dependent cell structure. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Artificial Neural Networks, pp. 403–408. Elsevier, Amsterdam (1991)
16. Cheng, Y.: Clustering with competing self-organizing maps. In: Proc.Int. Conf. Neural Networks, vol. 4, pp. 785–790 (1992)

17. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Francisco (1992)
18. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey (1984)
19. Utgoff, P.E., et al.: Decision tree induction based on efficient tree restructuring. Machine Learning 29(1), 5–44 (1997)
20. Basseto, B.A., Neto, J.J.: A stochastic musical composer based on adaptative algorithms. In: Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação, SBC 1999, PUC-RIO, Rio de Janeiro, Brazil, vol. 3, pp. 105–130 (July 1999)
21. Jackson, Q.T.: Adaptive predicates in natural language parsing. Perfection (4) (2000)
22. Costa, E.R., Hirakawa, A.R., Neto, J.J.: An adaptive alternative for syntactic pattern recognition. In: Proceeding of 3rd International Symposiumon Robotics and Automation, ISRA, Toluca, Mexico, pp. 409–413 (September 2002)
23. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
24. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst., Man, Cybern. 3, 28–44 (1973)
25. Takagi, T., Sugeno, M.: Derivation of fuzzy control rules from human operator's control actions. In: Proc. IFAC Symp. Fuzzy Inform., Knowledge Representation and Decision Analysis, pp. 55–60 (July 1983)
26. Lee, C.C.: Fuzzy logic in control systems: Fuzzy logic controller-Part I. IEEE Trans. Syst., Man, Cybern., 20, 404–418 (1990)
27. Papazoglou, M.P; et al, Service-Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings 05462 Service Oriented Computing (SOC) 2006
28. Patel, K.: Improvements on WSOL Grammar and Premier WSOL Parser. Research Report. SCE-03-25 (2003)
29. Yang, K., et al.: QoS-Aware Service Selection Algorithms for Pervasive Service Composition in Mobile Wireless Environments. Mobile Netw Appl. (2009)
30. Clement, L., et al.: UDDI Version 3.0.2, Tech. Rep., OASIS (2004), `http://uddi.org/pubs/uddi-v3.0.2-20041019.htm` (last access in January 2010)
31. Bottazzi, D., Montanari, R., Toninelli, A.: Context-Aware Middleware for Anytime, Anywhere Social Networks. IEEE Intelligent Systems 22(5), 23–32 (2007)
32. Gupta, A., Kalra, A., Boston, D., Borcea, C.: MobiSoC: A Middleware for Mobile Social Computing Applications. In: Mobilware 2009 (2009)
33. Bottazzi, D., Montanari, R., Giovanni, R.: A self-organizing group management middleware for mobile ad-hoc networks. Computer Communications 31, 3040–3048 (2008)
34. Cugola, G., Nitto, E.D.: On adopting Content-Based Routing in service-oriented architectures. Information and Software Technology 50, 22–35 (2008)
35. Yang, K., et al.: QoS-Aware Service Selection Algorithms for Pervasive Service Composition in Mobile Wireless Environments. Mobile Netw Appl. (2009)
36. Vesanto, J., Alhoniemi, E., Himberg, J., Parhankangas, J.: Som Toolbox 2.0 BETA online documentation (1999), `http://cis.hut.fi/projects/somtoolbox`
37. Free software available in, `http://wireless-matlab.sourceforge.net/` (last access in January 2010)
38. Free software available in, `http://www.cs.waikato.ac.nz/ml/weka` (last access in January 2010)