Matching Conflicts: Functional Validation of Agents*

George Cybenko and Guofei Jiang

Thayer School of Engineering, Dartmouth College
Hanover, NH 03755, USA
gvc@dartmouth.edu, guofei.jiang@dartmouth.edu

Abstract

In most working and proposed multiagent systems, the problem of identifying and locating agents that can provide specific services is a major problem of concern. A broker or matchmaker service is often proposed as a solution. These systems use keywords drawn from application domain ontologies to specify agent services, usually framed within some sort of knowledge representation language. However, we believe that keywords and ontologies cannot be defined and interpreted precisely enough to make brokering or matchmaking among agents sufficiently robust in a truly distributed, heterogeneous, multiagent computing environment. This creates *matching conflicts* between a client agent's requested functionality and a service agent's actual functionality. We propose a new form of interagent communication, called *functional validation,* specifically designed to solve such matching conflicts. In this paper we introduce the functional validation concept, analyze the possible situations that can arise in validation problems and formalize the mathematical framework around which further work can be done.

1.0 Introduction

Several efforts are underway to build multiagent *computational grids*, such as Globus(Foster and Kesselman 1998), Sciagents(Drashansky, Joshi and Riceand 1995), Infospheres(see http://www.infospheres.caltech.edu/). Such "grids" will be populated by server agents and services that are available to user-defined simulations and applications(Kotz et al. 1997). Computational resource agents will include distributed components such as databases, document repositories, mathematical software packages, sensor feeds, online simulations and even some hardware for computing. For example, a grid agent may implement a specific computational service such as solving specialized structured linear systems of equations or performing complex data transformations. Similarly, a grid agent may offer data products as a service, such as an ocean model performing real-time modeling that makes its state estimates available upon request.

*Copyright 1999 by AAAI Press. In the AAAI Workshop of Agent Conflicts, pages 14 -19, Orlando, Florida, July 1999. Available at http://agent.cs.dartmouth.edu/papers/.

A fundamental capability required in such a grid is a directory service or broker agent that dynamically matches user requirements with available resource agents. On the web, this capability is provided by search engines that index web pages and implement retrieval services. Whereas humans are typically the consumers of web information, grid agents will be the producers and consumers of grid resources with humans occasionally steering or interpreting the computations.

A grid agent that needs a computational service, such as for example the solution to structured linear system or a multidimensional Fourier transform, will locate the required service by consulting a distributed object request broker agent or a matchmaker service agent. For example, CORBA is an infrastructure for implementing distributed applications and provides a broker agent as a key component(Natan 1995). An object request broker agent (ORBA) not only locates a component or object that performs the required service but also mediates communications between the client agent and the service agent. In standard terminology, a matchmaker agent is an ORBA with reduced capability. A matchmaker agent merely locates remote agents or services but does not mediate communications between client and server agents. In the matchmaker agent framework, a client agent and the remote agent that it invokes communicate directly once their locations are made known by the matchmaker service agent.

2.0 The Functional Validation Problem

The server agents will advertise their services catalog on ORB's or matchmaker agents. Then just like web search engines, ORB's and matchmaker agents will need keywords and ontologies to specify agent services. Ontologies specify a domain and keywords specify functionality within that domain. For example, ontologies are envisioned for signal processing, ocean modeling, image processing, weather modeling and so on. Within an ontology, keywords such as "Fourier transform" and "linear system solver" will have possibly domain specific meanings. Several systems have been proposed for implementing such ontological matchings (see http://logic.Stanford.edu/kif/specification.html and http://www.cs.umbc.edu/kqml/).

Note however, there are literally dozens of different algorithms for implementing Discrete Fourier Transforms

(Brigham 1988). Different algorithms make different assumptions about the symmetries of the input vector and order the output in a variety of ways. Some algorithms may be only able to transform the input vector of some certain dimensions. The actual numerical computations carried out vary from algorithm to algorithm so that different round-off errors are accumulated leading to slightly different answers. Moreover different numerical implementations of some basic computations in an algorithm such as Integration and Derivation always lead to different computational speed, different accuracy and so on. The same is true of linear system solvers, other numerical algorithms and data products. Especially in some complicated computation tasks, the possible situations are more challenging. Fox example, there are a lot of different system modeling algorithms developed for different control systems such as ARMX or ARMAX system, time variant or invariant system, noisy or non-noisy system, linear or nonlinear system, and so on. So in this case it's more difficult for the keywords and ontologies to precisely describe the real functionality of the algorithms.

Keywords and ontologies cannot be defined and interpreted precisely enough to make brokering or matchmaking between grid agents' services robust in a truly distributed, heterogeneous computing environment. This is the basis for *matching conflicts* between client agents' requests and service agents' responses. Some form of functional validation of resource agents will be required.

Functional validation means that a client agent presents to a prospective service agent a sequence of *challenges*. The service agent replies to these challenges with corresponding *answers*. Only after the client agent is satisfied that the service agent's answers are consistent with the client agent's expectations is an actual commitment made to using the service. This is especially important in mission critical applications. In fact we can find the same idea of functional validation in our daily life. For example, a demo is often used to show the functionality of some software.

Our ongoing research on agent-based systems (see http://actcomm.dartmouth.edu and http://www.cs.dartmouth.edu/~agent/) has led us to the conclusion that semantic brokering will not be sufficient to implement truly distributed, heterogeneous multiagent computing. Two steps are required before agents commit to each other:

1. Service agent *identification* and *location*;
2. Service agent *functional validation*.
3. *Commitment* to the service agent.

These steps are shown in Figure 1. Identification and location will be performed by ORBA's or matchmaker agents and is already an area of active research. However, functional validation of distributed components and agents is a new subject of research that is essential for the future success of truly heterogeneous, distributed computing grids.

3.0 An Example

Suppose a grid resource agent has been developed to model and predict a regional ocean circulation. It requires a variety of grid-based, distributed data products (measurement results for example) and a three-dimensional Discrete Fourier Transform to implement a spectral method. The grid computation is dynamic - it uses the best available resources at any time.

During runtime, a request is made to an ORBA or a matchmaker agent for an appropriate DFT. This request is made based on keywords and possibly parameter lists for invoking the remote agent. The ORBA or matchmaker agent retrievals its service catalog and returns with several candidate remote agents and specifies their returning value structures. This is Step 1. outlined above.

With this information, the ocean circulation model must validate the functionality of the remote DFT agent, typically written, maintained and updated by others. This validation is done at the beginning of the run or whenever a previously validated agent has failed and a new agent must be located to continue operation. *This functional validation is entirely different from issues of authentication and certification. The performance and correctness of the remote agents' service may have been authenticated and certified by an authoritative procedure or person. However, the actual functionality of the remote agent must be validated before the client agent's ocean simulation commits to using it. This is because what was "correct" and "sufficient" in the eyes of the certifying authority may not be enough for the client agent to conclude that the service is "correct". Moreover, the keyword description of the service may be incomplete or inconclusive in the eyes of the client process.*

The fundamental question asked by agent functional validation is:

> *Do all parties involved in the computation agree on the actual agent functionality?*

Is the functionality of the remote DFT agent what the simulation agent requested? This cannot be answered by keyword matching or certification alone.

Our approach to functional validation is to allow the client agent to challenge the service agent with test cases, $x_1, x_2, \ldots, x_k$. The service agent provides corresponding responses/answers, $f_R(x_1), f_R(x_2), \ldots, f_R(x_k)$. The client agent may or may not have independent access to the correct responses/answers, $f_C(x_1), f_C(x_2), \ldots, f_C(x_k)$. Depending on the sequence of responses, the client agent may or may not commit to using (and therefore possibly paying) the service agent. To implement such agent functional validation, several questions need answering:

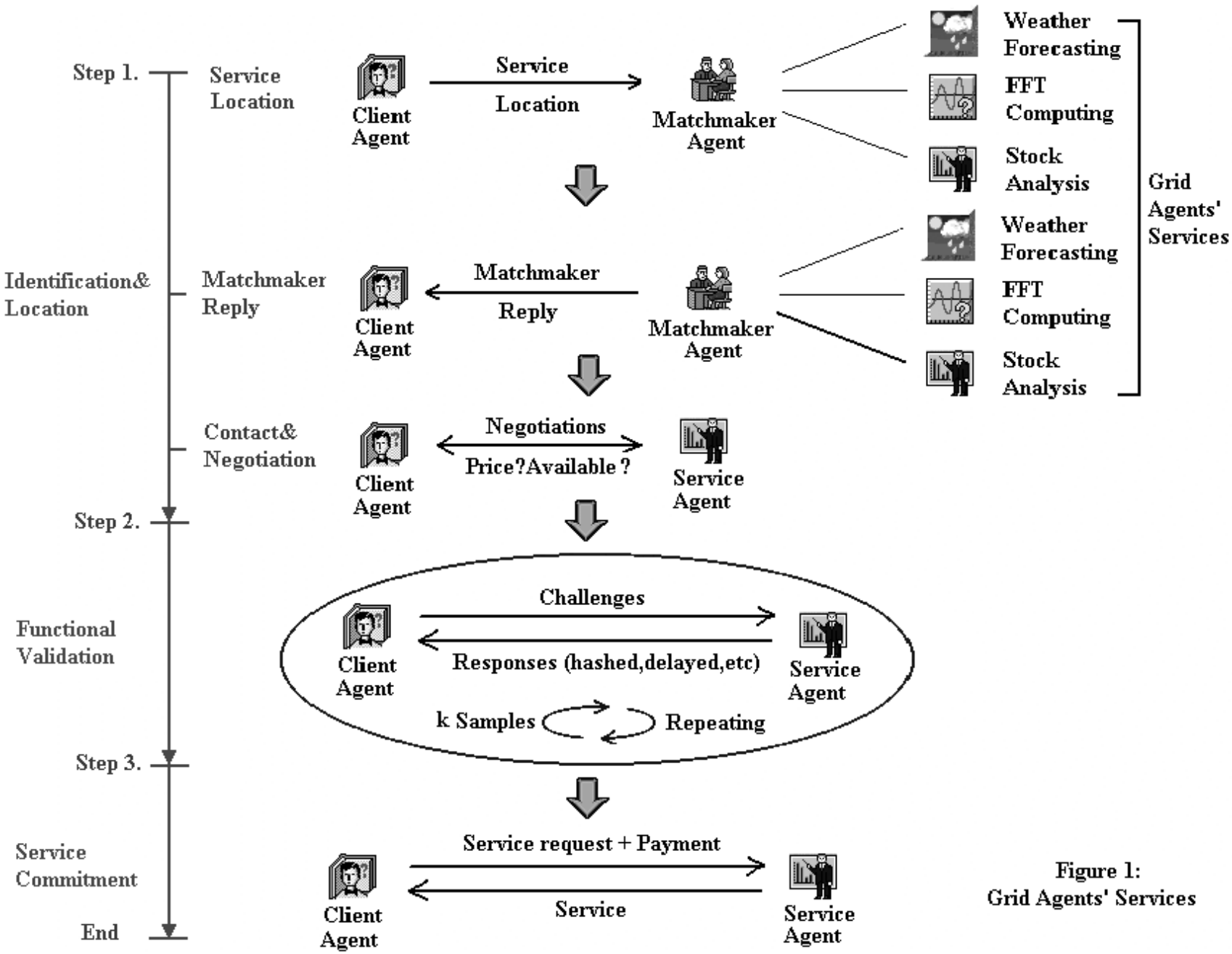

Figure 1:
Grid Agents' Services

- How large should k be?
- What if $f_C(x)$ is not known by the requesting client agent?
- What if the service agent is fee based and so answers, $f_R(x)$, are not given away freely before commitment?
- How to implement the validation process online?

These questions lead to important, novel mathematical investigations into functional validation.

4.0 Possible Agent Functional Validation Situations

We will formalize the functional validation program for a computational and data service as follows. Denote the client agent's calling simulation by ***C*** and the remote agent's service component by ***R***. ***C*** requires the evaluation of a function, $f(x)$, where x is the input parameter. Assuming compatibility of input and output parameter structures and types, which has already been checked by ORBA or matchmaker agent's services, the remote agent's service is expected to provide $f_R(x)$. There are several possible situations that can arise.

4.1 *C* "knows" $f_C(x)$ and *R* provides $f_R(x)$

This is the simplest case. Here the word "knows" means that ***C*** itself is able to know $f_C(x)$ of *the selected samples*. For example, the client agent, ***C***, needs to complete a complicated computation such as a huge dimension matrix inverse operation. It assumes that ***C*** has precomputed or otherwise correct responses $f_C(x)$ for some simple test cases such as some low dimension matrix inverse operations (which are easy to be computed by the client agent itself), and that the service agent, ***R***, provides responses $f_R(x)$. ***C*** challenges ***R*** with these simple sample inputs. After the service results, $f_R(x)$, are returned, we must determine whether ***R*** is implementing the "correct" service by comparing $f_R(x)$ and $f_C(x)$.

Basically we can formalize these problems and answer them using PAC (Probably Approximately Correct) Learning Theory (Kearns and Vazirani 1994) as a starting point. We believe that the semantic level brokering can be used to achieve agreement about which function class the client and service functions belong to. Traditional software testing (Beizer 1990) and certification by other experts can be viewed as verification that the service belongs to the function class. Functional validation establishes a high level of confidence that the actual functions themselves match.

4.2 *C* "knows" $f_C(x)$ but *R* does not provide $f_R(x)$

In the above case, the remote service agent, ***R***, may not be willing to offer the exact results for the challenges that ***C*** poses. This is the case when, for example, ***R*** is a fee-based service and cannot be sure whether ***C*** will use these free responses in its actual application but not for validation purposes. So service agent ***R*** could offer hashed results, $g(f_R(x))$, where g is a secure hash function specified by ***C***, ***R*** or an intermediary. Fox example, $g(f_R(x)) = Y ? f_R(x)$, where Y is a singular matrix. Because Y^{-1} doesn't exist, ***C*** will not be able to compute $f_R(x)$ out from the returned $g(f_R(x))$ and Y, i.e. $Y^{-1} g(f_R(x)) = Y^{-1} ? Y ? f_R(x) = f_R(x)$.

By comparing $g(f_C(x))$ and $g(f_R(x))$, we can formalize this case in the same way as 4.1 using ideas from Zero Knowledge Proof theory (Goldreich and Oren 1994). Zero Knowledge Proofs have been used to securely exchange information between corresponding parties without giving away any unnecessary knowledge. For example, the challenge here would be to "prove" that a service agent "knows" how to compute DFT's without actually giving away any information that could be useful to the client agent presenting the challenges.

4.3 *C* does not "know" $f_C(x)$ but *R* does provide $f_R(x)$

In the above, we have discussed situations where ***C*** "knows" $f_C(x)$. In fact, in some cases ***C*** itself will not be able to "know" $f_C(x)$. But sometimes ***C*** may be able to verify ***R***'s service in an "indirect" way.

For example, ***C*** may request weather forecast data from a service agent, ***R***. ***C*** cannot verify ***R***'s responses without a time delay, namely recording the predictions and evaluating them at the time that the actual weather is known. This is a simple case of "simulation-based" validation. Let's think about the matrix inverse operation example again. Now assuming that ***C*** even doesn't know $f_C(x) = x^{-1}$ of the test samples, ***C*** can still verify ***R***'s service by multiplying the input matrix x with the returned $f_R(x)$ and checking whether $x ? f_R(x)$ is equal to an identity or not.

In an more complex example, suppose the calling component, ***C***, is controlling a plant with some internal states unobservable by ***C*** directly. Based on the observed state and time solely, ***C*** is not able to observe or validate directly the internal states or parameters, $f_C(x)$, of the plant. Instead, it consults a system identification service from the remote agent ***R***. By using the returned parameters, $f_R(x)$, in its control policy and observing the plant's performance, then ***C*** can try to verify whether ***R*** is offering the desired service by evaluating its own control performance.

In this case, basically we will try to employ simulation-based learning, reinforcement learning, and unsupervised learning techniques for functional validation (Jiang 1998).

4.4 *C* does not "know" $f_C(x)$ and *R* does not provide $f_R(x)$

This case can arise, for example, when client agent ***C*** requires a certain service but the service agent ***R*** provides a related (derived or hashed values, for example) service. This is the most difficult situation. For the matrix inverse operation example here, ***C*** doesn't know what $f_C(x)$ is and just like in case 4.2, ***R*** may be only willing to offer the hashed result $g(f_R(x)) = x^{-1} ?Y$. But even in this case, like the validation approach in the case 4.3, ***C*** can still verify ***R***'s service by checking whether $x ?g(f_R(x))$ is equal to $x ?x^{-1} ?Y = Y$ or not.

So basically we believe this situation can be reduced to a combination of 4.2 and 4.3.

5.0 A Mathematical Model for Agent Validation – PAC Learning

In general, we can formalize the problems arising in the above four situations and answer them using PAC learning theory. Here the goal of PAC learning is to use as few examples as possible, and as little computation as possible to pick a hypothesis concept which is a close approximation to the target concept. Then by the PAC learned hypothesis, the client agent can conclude whether the service agent can offer the "correct" service.

For convenience, let's consider the simplest cases in situation 4.1. Here assume the input space X is a fixed set, either finite, countably infinite, $[0,1]^n$, or E^n (Euclidean *n*-dimensional space) for some $n ? 1$. For the agent functional validation problem, we are concerned with whether the service agent can offer the "correct" service, so we define a concept to be a boolean mapping $c : X \blacklozenge \{0,1\}$, with $c(x) = 1$ indicating that *x* is a positive example of *c*, i.e. the service agent provides the "correct" service for challenge *x*, and $c(x) = 0$ indicating that *x* is a negative example, i.e. the service agent does not offer the "correct" service for challenge *x*. A concept class C over X is a collection of concepts *c* over X. So here the single unknown target concept is either that the service agent does offer the "correct" service, i.e.

$$\|f_C(x) - f_R(x)\|_X \le \gamma$$

(where γ is the allowable computational error), or that the service agent does not offer the "correct" service, i.e. $\|f_C(x) - f_R(x)\|_X ? \gamma$.

Let P be a fixed probability distribution on X and assume that examples are created by drawing challenges independently and randomly according to P. Define an index function

$$F(x) = \begin{cases} 1 & \text{if } \|f_C(x) - f_R(x)\| \le \gamma \\ 0 & \text{otherwise} \end{cases}$$

Then the client agent can randomly pick m samples $S_m = \{(x_1, F(x_1)), (x_2, F(x_2)), \cdots, (x_m, F(x_m))\}$ to learn a hypothesis $h \subseteq H$ about whether the service agent can offer the "correct" service, where H is the hypothesis space and usually is the concept class C itself. Here let $A_{C,H}$ denote the set of all learning functions $A : S_m \blacklozenge H$. We claim that $A \subseteq A_{C,H}$ is consistent if its hypothesis always agrees with the samples, that is $h = A(S_m)$. Thus based on the PAC learned hypothesis which is a close approximation to the real target concept, the client agent can conclude whether the service agent can offer the "correct" service with some confidence.

Now the problem is that how many samples or challenges are needed to conclude a hypothesis that is enough approximate to the real target concept. Define the error between the target concept c and the hypothesis h as

$$error(h) = \Pr ob_{x \subseteq P}[c(x) ? h(x)]$$

where $\Pr ob_{x \subseteq P}[.]$ indicates the probability with respect to the random draw of x according to P. Then mathematically we can formalize the above problem as follows: How large must the number of challenges, m, be so that

$$\Pr ob^m\{error(h) \le \varepsilon\} ? 1 - \delta$$

where ε is the accuracy parameter and δ is the confidence parameter.

Blumer et.al (1989) solved this problem with the following powerful result

Theorem (Blumer et al. 1989) *Let H be any well-behaved hypothesis space of finite VC dimension d contained in 2^X, P be any probability distribution on X and the target concept c be any Borel set contained in X. Then for any $0 < \varepsilon, \delta < 1$, given*

$$m ? \max\left(\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{13}{\varepsilon}\right)$$

independent random examples of c drawn according to P, with probability at least $1-\delta$, every hypothesis in H that is consistent with all of these examples has error at most ε.

In the above theorem, the Vapnik-Chervonenkis dimension (VC dimension) is a combinatorial measure of

concept class complexity which assigns to each concept class C a single number that characterizes the sample size needed to PAC learn C. See its detailed definition in (Blumer et al. 1989).

Thus by determining the boolean concept's VC dimension over X and selecting an accuracy parameter ε and a confidence parameter δ, according to the above theorem, the client agent can pick m samples to PAC learn a hypothesis which is close enough to the real target concept, i.e. whether the service agent can offer the "correct" services. However, with regard to special situations in the agent functional validation problem, for example, sometimes one negative example is enough to say that the service agent can not offer the "correct" service, we believe that we can get some better results by further work on this problem.

6.0 Conclusions

In a multiagent grid, a broker or matchmaker agent will use keywords and ontologies to specify grid services. However keywords and ontologies cannot be defined and interpreted precisely enough to make brokering or matchmaking between resource agent services sufficiently robust in a truly distributed, heterogeneous computing environment. This creates *matching conflicts* between client agents and service agents. Agent functional validation is proposed and studied in this paper. It appears to be promising approach to resolve matching conflicts in distributed multiagent grids.

Acknowledgements: This work was partially supported by Air Force Office of Scientific Research grants F49620-97-1-0382, National Science Foundation grant CCR-9813744 and DARPA contract F30602-98-2-0107.

References

Beizer, B., 1990. *Software Testing Techniques*, Van Nostrand Reinhold, New York.

Blumer, A. et al., 1989. Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the Assocation for Computing Machinery* 36(4):929.

Brigham, E.O., 1988. *The Fast Fourier Transform and Its Applications,* Prentice Hall, Englewood Cliffs, New Jersey.

Drashansky, T.T., Joshi, A. and Rice, J.R. 1995. SciAgents – An agent based environment for distributed, cooperative scientific computing. In Proc. Of the 7th International Conference on Tools with Artificial Intelligence. IEEE Computer Society, 452-459.

Foster, I., and Kesselman, C., 1998. *The Grid Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, San Francisco, California.

Goldreich, O., and Oren. Y.,1994. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology* 7(1): 1.

Jiang, G., 1998. Reinforcement Learning Methods Based on Dynamic Programming, Ph.D. thesis, Department of Automatic Control, Beijing Institute of Technology.

Kearns, M.J., and Vazirani, U.V., 1994. *An Introduction to Computational Learning Theory,* The MIT Press, Cambridge, Massachusetts.

Kotz,D., Gray,R., Nog,s., Rus D., Chawla,S. and Cybenko,G. Agent Tcl: Targeting the needs of mobile computers. *IEEE Internet Computing*, 1(4):58.

Natan R.B.,1995. *Corba-A Guide to Common Object Request Broker Architecture*, McGraw-Hill, New York.