

A Personalized Active Services and Decision-making System —— MultiAgent-Based Data Mining Approach

Hui Li Zhuoqun Xu NingZhang

Computer Science and Technology Department, Peking University Beijing, China, 100871
Tel: 86-10-62751798 E-mail: lihui@ailab.pku.edu.cn

ABSTRACT OracularPlan is a Multi-Agent System for the purpose of bettering macroscopical management/decision-making in the distributed and open environment, and at the same time, meet personal requirements in real time. Based upon the system, data mining and machine learning can be used to acquire useful pattern and significant predictive model in order to provide more reliable services and provide more precise decision-making. For example, the predictions of travel time in an automobile. For dynamic changes of the traffic condition, we fail to get the accurate prediction from existing digital maps. In addition, drivers don't know the conditions with each other, so, when there are 2 paths available, they may choice the same one. As a result, the traffic congestion may happen.

In our system, we can analyze 2 kinds of data. One is the information about the whole process of a user's decision-making. With the analysis, we induce the general users' behavior pattern. So, the system can predict the user's activity and thus better meet his need. The second one is the geographical and temporal data. The system provides a set of learning (prediction) programs to compute the model over the data distributed geographically. It also provide a set of strategies, called Meta learning (prediction) to help distributed Data Mining method to cooperate for the more precise prediction. It is proved that the general model is more adaptive than the single model in the dynamic environment. Our system provides such a convenience to design one strategy to smooth the collision between personal requirement and macroscopical management. One of our system target applications is transportation services and traffic management. The system can help drivers to find the optimal path, and to improve overall traffic flow.

KEYWORDS Multi-Agent System, KQML, KIF, PMML, Data Mining

1. Introduction

Combining Data WareHouse and Datamining is one popular method to cons DSS constructing method is. Data WareHouse collects the data located at distributed sites. Then, the volume history data are analysis by using various kinds of Data Mining methods.

Nowadays, there is more and more decision-making requirement in the outside and dynamic environment. There are such a kinds of application, in which there are two kinds of requirement, which relate with each other and even conflict with each other, and we still have to make decision in real time. For example, in the transportation management, usually the drivers want to know which route is the shortest one from one site to another one, the requirement is called Micro-Requirement; at the same time, the management department of the transport system hopes that the traffic congestion don't happen and traffic flow is distributed reasonably in the transport system, the requirement is called Macro-Requirement. So, in this application, there are 2 kinds of decisions, one for personal level, one for system level. Specially, the former need to response in real time. If we construct the decision support using Data warehouse, then obtaining the real time data will be a bottleneck when we want to get decision-making in real time for the confinement

of the network transportation. On the other hand, in many cases, data are inherently distributed and cannot be accessed from any one machine. So, for the solution of the problem shown above, the system must provide the following advantage:

- Support the ability to learn the predictive model in the distributed, dynamic changing, open environment;
- Support the interoperation between the heterogeneous resource;
- Support the system's adaptability to the environment.

If so, we can analyze the data stored in the distributed database. And the any part in the system can access the data or analysis result in the whole system without necessarily knowing where they are in the system. For only the analysis result is transported, real-time requirement will be improved.

Here, we present one system, called OracularPlan. It provides reasonable system architecture to meet the challenge showed above. First, we need to have a deeper understanding of what is data, what is the underlying disciplinarian under the distributed, dynamically changing, open environment. So, we get the understanding of the resource from the semantic level, and have a higher view over the resource in the distributed and open environment. In the system, individual

requirement, independent of the identity of the user, the time of request, where the requested information is, and what its' structure is, is analyzed and interpreted, and automatically dispatched to the corresponding execution unit. For improving the precision of the analysis, these execution units can cooperate with each other according to the context given by the system semantic level, so the interoperation is implemented. Furthermore, the system also provide the ability to record the behavior of individual request, and analysis its' behavior pattern for active service. In addition, from the view of the whole system, we also can analyze the behavior pattern of the confluence so that the context of the system can be revised to adapt the open environment.

To achieve the design purpose of OracularPlan, we integrate several newly developing technologies.

- **Agent Technology** Autonomous agents are computer systems that are capable of independent action in dynamic, unpredictable environments. Agents are also one of the most important and exciting areas of research and development in computer science today.
- **Ontology** Ontology is defined as a specification of a "conceptualization" in the AI community. Here, Ontology give a concise, uniform, and declarative description of semantic information, independent of the underlying syntactic representation or the conception mole of the open environment.
- **Network Computing** Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language. It is a good tool to carry out networking computing.. The main component in the system is written using Java, for example, various kinds of Agents, and the user interface.

In this paper, we present the architecture of the system OracularPlan and its' implementation. And also describes how to use the system to meet with our application.

The paper is organized as several sections shown below. In section 2, the overall architecture is presented. This design and implementation of the system is given in section 3. Section 4 describes the organization of the Agent Space and relationship between agents. Section 5 briefly gives one decision-making application in the distributed and open environment. In the last section, we give our opinion on the system, and state our future work.

2. Architecture

2.1. Architecture Overview

OracularPlan is a network system, which is used to data retrieval, data analysis and decision-making by means of the cooperation between Agents. Specially, it suits real-time decision-making application under the distributed, dynamically changing and open environment. In the distributed environment, the user's behavior can be recorded in two ways. One is that the system can record the individual behavior. The second is that the system can record the macroscopically effect reflecting many individual behavior. When one user sends out a request, the request under domain knowledge is received by a User Agent, which is only corresponding to the user in the agent running space. The request is analyzed and interpreted, then the User Agent send the message to one Task Agent by means of the context information provided by the Ontology Server. The message often is coded using KQML, an agent communication language. The User Agent also can communicate with the user and transfers the domain knowledge from the Ontology Server. In the user end, one Java Applet is served as the user interface. The independence-system of Java Applet benefits for users' communicating with the system anywhere. User Agent also is in charge of recording the information during the lifecycle of it's own. Furthermore, when the data is accumulated to one degree, the behavior pattern of the user can be analyzed. So, different users are corresponding to different User Agents with different characters. The User Agent is called Individuation User Agent. The Task Agent mentioned above first enter the agent space, it registers its service in the Facilitator, one special Agent. The service can be used in network in the transparent manner. It is in charge of the whole process of implementing the task. The Task Agents also can cooperate with each other according to the certain the agent context. All the kinds of the Agents in the system are composed of one Agent Space. The Agent Space Administrator and Ontology Server manage the Agent Space. The system architecture is shown in the Fig. 1.

The functionality of main components, including all kinds of Agents is briefly described below. We will give more detailed description in the next section.

Agent Space Administrator: It consists of 2 parts, i.e., Agent component base and Agent persistent object base. The former stores General Agent Component and some ad hoc Agent Component. The latter stores the inactive User Agent with individuation. It also provides many

services.

Ontology Server: Providing the conception and description of the system, outside environment and Agents. And also provides the require services.

Knowledge Base: Storing the domain

knowledge.

User Agent: Corresponding to the user that has registered in the system. It also the only entry the

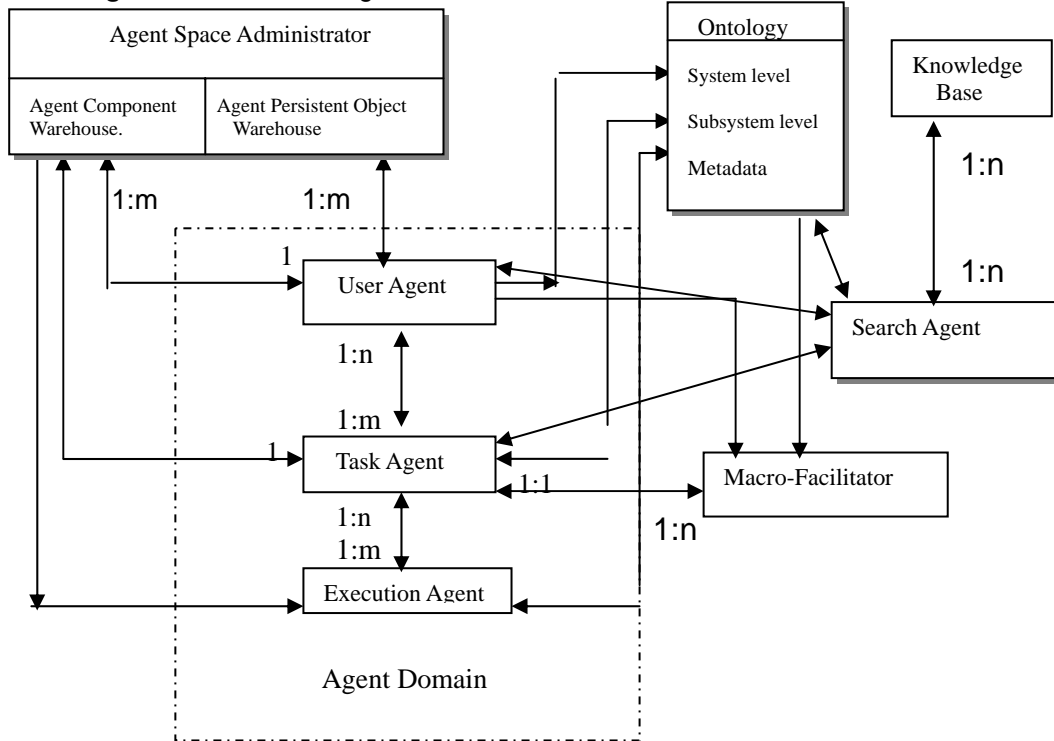


Fig 1: The OracularPlan architecture

user send out requests to the system. It knows Which Task Agent can meets with the request by means of the Ontology Server.

Facilitator: A facilitator is an agent that performs various useful communication services, e.g. maintaining a registry of service names, forwarding messages to named services, routing messages based on content, providing "matchmaking" between information providers and clients, and providing mediation and translation services.

Task Agent: Be in charge of the implementation of one task. It analyzes the user's request, and divides it into several sub-tasks according to the knowledge from the Ontology Server. It also integrates the results from the sub-task execution unit into one. One task can be considered one decision-making or one requires.

Execution Agent: Special Agent for implementation of tasks. It consists of Search Agent and DataMing Agent, etc. Search Agent takes charge of information retrieval. It can commit Facilitator to find the available Agent that knows where information is. DataMining Agent is in charge of data analysis and prediction in the distributed environment. It will take an important

role in the distributed decision-making application. The DataMining Agent in OracularPlan can analyze data located at different sites and choice the best predictive model. They also can communicate with each other for the better predictive model.

2.2. The Communication and Knowledge Sharing within Multi-Agent System

Agent Communication Language (ACL) is one kind of high-level network communication language. KQML is one of them. In addition, the content in KQML is the knowledge sharing part. Here, introduce 2 kinds of language, KIF and PMML.

> KQML

KQML or the Knowledge Query and Manipulation Language is a language and protocol for exchanging information and knowledge. It is part of a larger effort, the ARPA Knowledge Sharing Effort that is aimed at developing techniques and methodology for building large-scale knowledge bases, which are sharable and reusable. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an

intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving.

➤ KIF

KIF is a computer-oriented language for the interchange of knowledge among disparate programs. It has declarative semantics, i.e., the meaning of the expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions; it is Logically comprehensive, i.e., express the arbitrary sentence using 1st-order predicate calculus. It provides the representation of knowledge about representation of knowledge. It provides for the representation of nonmonotonic reasoning rules. It also provides the definition of objects, functions, and relations.

➤ PMML

In the OraclePlan, there is also one Language, PMML. Predictive Model Markup Language (PMML) is an XML-based language which provides a quick and easy way for companies to define predictive models and share models between compliant vendors' applications. The Data Mining Group ratified the latest version of PMML on July 13, 1999.

In the system, an agent can transform data or queries from the local internal format into KIF or PMML, then wraps the KIF or PMML message in a KQML performative.

3 Design and Implementation

In this section, we describe the design and implementation of the main components in the system OraclePlan.

3.1. Agent Space Administrator

This subsystem includes 2 parts, Agent Component Base and Agent Persistent Object Base. In addition, also provide the administration function, such as register, edit, delete, etc.

In the Agent Component Base, all the information about Agent is stored. When the system start up, they also start up. The Agents in the Agent Space is their instances, or ones of their subclasses. The Agent Persistent Object Base stores the inactive User Agent with individuation, which have ever registered in the Base. The history data of User Agent are recorded.

3.2. General Agent

We choice Java to construct the system. We first construct an abstract class GAgent, the instance of which is a general Agent. All the other Agents is its inherit class. In the general agent, We implement message handler, action handler, agenda handler, etc. Message handler manages the message queue, and is in charge of send, receive and interpret

messages. The inter-agent language is KQML. Action handler handles message and executes applications in agent, or deal with message or data stored at specified network location. Agenda handler deal with implementation of special task, for example the decision task or query tasks. When communicating with other agents, Agent itself doesn't pause its other action for multi-thread process inside agent. For Java's support of multi-thread, asynchronous operations between agents are easier. Java is an independent-system language, so it is easy to implement the mobile character of Agent. In addition, Java provides the function of clone. In conclusion, with application of Java, it is easier to implement the characters of Agent, i.e., autonomous, asynchronous operations, mobile, clone, etc.

In the following section, we introduce several inheritance agent class of general class.

3.3. User Agent

3.3.1. Function and Design

User Agent is not only the interface between one user and the system, but also the representative of the user in the Agent computing space. It receives the user's queries, and choices available Facilitator and Task Agent for it by means of Ontology Server. When the lifecycle of the User Agent ends, it is saved into Agent Persistent Object Base. User Agent isn't passive "request-response", and it can provide services for the users actively, especially in the dynamic and open environment.

3.3.2. Implementation

User Agent is running in the Agent Working Space. It communicates with user with Java Applet. So, User Agent can give the User a view of Ontology in the user domain. That means, users can choice in which state of the system he wants to work with the Multi-Agent System. So, user can tailor the query effect and decision effect. The communication protocol between Java Applet and User Agent adopts RMI or IP (KQML wrapped in it). In addition, the advantage Java Applet in the system provides users with flexible, independent-system, and context-sensitive user interface. The character is more suitable in the situation where application environment is distributed widely and dynamically changing. User Agent is also in charge of the analysis of user request in virtue of the domain knowledge, and dispatches the request to the corresponding Task Agent. It also is acted as the Communication Bridge between the Task Agent and Java Applet.

When the lifecycle of User Agent ends, it saves

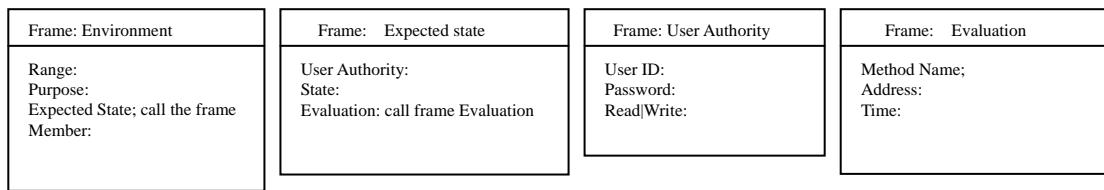


Fig 2 the Environment Conception Model

the context information and running information into Agent Persistent Base with JDBC/ODBC. Then, it withdraws this running.

3.4. Task Agent

Task Agent receives task requests from the User Agent. First, it determines the description and the decomposition of the task then give the execution strategy. All the process is depend on the domain knowledge from Ontology Server. And then, it interacts with Agent Administrator and Search Agent for obtaining the necessary resource. It also harmonizes the agents that are running for the task and integrates the result from sub-task execution units. Task Agent is designed to deal with dynamic, incomplete and uncertain knowledge. For this purpose, declarative task plans are given; Task Agent based upon which can adopt a kind of general form to take part in task plan. Below, we give one method to implement task plan.

First, in the system, there are many sub-execution nodes as plan node. They are reusable component. Then, constructing one Rule-based Knowledge Base, in which the logic relations between these plan nodes are given. For example, the relation between 2 nodes is partial-order or full-order, and these relation and even its precondition are recorded in the Knowledge Base. When one request is received, Task Agent first transforms the request form so that its form is consistent with the form that the Knowledge Base can deduce. Then, searching the Knowledge Base, and get a set of sub-execution node and logic relation among them. Following, Task Agent send out requests to Facilitator, and each request represent one request of the implementation of sub-execution node. The Facilitator knows their context information, so their running logic is known. When implementing the process, the multi-thread mechanism, and the services provided by Agent itself, such as message-handler, action-handler, open-server-handler and agenda-handler will make it easier. We will present one application in the latter section using the method.

3.5. DataMining Agent

DataMining Agent is in charge of data analysis and prediction. So, in addition to the function of general agent, it also has some built-in data-mining methods. It can be created in 2 ways. One is done in run time. The second one is always running and waiting to the request. It is proved that, in the dynamical environment, the predictive model is changeable. And this kind change is related with other predictive model located at other sites. For more precise prediction, DataMining Agent needs to know other Agent's model. PMML makes it easier. PMML describe the data schema, the name of predictive model, the type of predictive model, test data name, measurement of test data, the attribute predicted, the description of all the nodes in the prediction process. DataMining Agent can wrap PMML into IP, and broadcast it to all concerned DataMining Agent.

3.6. Ontology Server

Ontology is a philosophy conception. In AI community, usually it is defined as the specification of conceptualization. Here, we give our understanding. In the distributed, dynamically changing and open environment, Ontology should have the characters shown below.

- Describe the infrastructure of data, which can be recorded to describe anything in the world. It makes interoperation possible between any components.
- Describe the conception of the multi-level environment. It tells us the system should be running under the state which both objective confinement and subjective requirement control.
- The infrastructure of Multi-Agent System. It indicates the context of Agents when it is running.

To a), the MetaData architecture meets with it. Lots of researchers and organizations study it. To c), we discuss a lot above. So, here, we only talk about b).

We describe multi-level environment using the frame structure. It can represent attributes from

various aspects of thing, the relationship among things, the characters of things, differentiation, etc. Here, we define an outside environment users reside in using frame (Fig 2).

Among which, the Frame Environment represents the outside environment our system describes. Name is the name of the environment; Range is the range of the environment; Purpose is the field of the application; Expected State is a frame, defined as the expected state of the environment, and which is a subjective conception in some sense; Member is a set, in which every element is one Environment Frame. In the Frame Expected State, User Authority is a frame, which describes who can access the state of the environment; State is a description of the system state, which can be defined in the Metadata; Evaluation is also a frame, which describes the Evaluation Method and where it is stored, register time.

4. Organization of Agent Space and Relationship between Agents

4.1. Organization of Agent Space

For the management in the Agent Space, we divide the Agents into different domain with Ontology_id and Task_id. As statement in the section above, every request from user is concerned about one concrete field. So, Ontology_id identifies the implementation process of the request. So, the Agents with the same Ontology_id belong to the same Agent Domain. In this domain, several User Agents may process different tasks. So, the User Agents with Task_id corresponding to Task Agent and all the other Agents involved in this task constitute a sub-Agent Domain. This structure benefits knowledge sharing both among the same Agent Domain and different Agent Domain

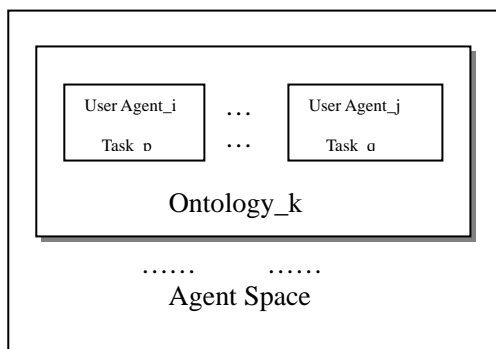


Fig 3 Organization of Agent Space

4.2. Relationship between Agents

Below, we give a concrete view to show how the

communicate between Agents work. Here, we only choice the relationship between User Agent and Task Agent.

After User Agent enters Agent Space, it determines to which Task Agent it send out request by means of the Agent context information. Then, it set up relation with Task Agent by sending out the request "ask-all" wrapped in KQML. When Task Agent first enter the Agent running Space, it registers its services to Facilitator by sending out message "Tell" wrapped in KQML. So, User Agent can access the services transparently in the network by sending message to Facilitator. In the Fig 4, two Advertisements, Tell and Ask-All are shown. They all are KQML message in which there is KIF query.

In "Tell" advertisement, Task Agent tells Facilitator that it can deal with the Minimum Time Route Problem within Beijing. In "ask-all" advertisement, User Agent ask that where are the Agents who know the Minimum Time Route from Zhongguancun to Tiananmen. Facilitator match the 2 message, and return the addresses, which meet with condition. After Task Agent receives the User Agent's request, it loads corresponding process program

5. Application

We have established one prototype system for running in the distributed and open environment. Now, we briefly present one application dealing with the management of transportation in Beijing.

First, we establish one conception of transportation environment in Beijing (it is only rough one, but can state some problems). The initialization of the "Member" field isn't written here. From this environment conception, we can know how many resource of the transportation infrastructure in Beijing can be used. In the frame, the attribute "State" is good, which means more than 90% resource available. In contrast, if "state" is very bad, that means that martial law is enforced. Here, the Onyology_ID is the Ontology_BJ_trasportation_1, with which the agents with Ontology_BJ_trasportation_1 can access domain knowledge.

Assuming 2 groups of drivers in the street. One group wants to find the minimum distant way home, the other one wants find a route with minimum time. So, their queries have different Task_ID, task_1 and task_2 respectievly. After they sends out queries, they are received by different User Agent. These User Agents can be divided into 2 groups, one identified by task_1, the other task_2. Assuming the corresponding Agent Domain is domain_1 and domain_2 respectively. By means of the context of Agent (which can be accessed from the Knowledge

Base with 2 identifies Ontology_BJ_transportation_1 and task_1), the User Agent can send out messages to corresponding Task Agent through Facilitator. Assuming they are Tagent_1 and Tagent_2. Tagent_1 understand what the query “minimum distant way” means, for it can access the plan of the query from the knowledge base. So, it make use of

Beijing map to calculate the dynamically. by using Classloader minimum distance way. For the task_1 isn't concerned about conflict problem, Tagent_1 returns the result to User Agent, and User Agent return User. The process is a database query problem in essential.

But, to the second query, it is far more complicated. For getting travel time needs the traffic

```
(Tell
: Sender "Task Agent"
: reciever "Faciliator"
:ontology "OracularPlan"
:language KIF
:content
  (and
    (agent ?a)(name ?a "TaskAgent1")
    (address ?addr)(address ?a ?addr)
    (host ?addr "plan.edu")(port?addr ?5001)
    (protocol ?addr "http")(type ?a ?TaskAgent)
    (language ?a ESQ)(ontology ?o)
    (ontology ?a ?o)(name ?o ? "Plan")
    (frame ?f)(name ? f? RoutePlan)
    (slot ? route)(slot ?f ?route)
    (slot ? map ) (slot ? f ?map)
    (constrain ?c)(constrain ?route ?c)
    (expression ? c(and
      (= ?route MINITIMERROUTE)
      (= ?map BeiJingMap)
    )
  )
)

(ask-all
:sender "User Agent"
:reciever "Faciliator"
:ontology "OracularPlan"
:language KIF
:aspect(?name ?protocol ?host ?port)
:content
  (and
    (agent ?a)(name ?a ?name)
    (address ?addr)(address ?a ?addr)
    (host ?addr ?host)(port?addr ?port)
    (protocol ?addr protocol)(type ?a ?TaskAgent)
    (language ?a ESQ)(ontology ?o)
    (ontology ?a ?o)(name ?o ? "Plan")
    (frame ?f)(name ? f? RoutePlan)
    (slot ? route)(slot ?f ?route)
    (slot ? map ) (slot ? f ?map)
    (slot ? from)(slot ?f ?from)
    (slot ? to ) (slot ? f ?to)
    (constrain ?c)(constrain ?route ?c)
    (expression ? c(and
      (= ?route MINITIMERROUTE)
      (= ?from Zhongguancun)
      (= ?to TianAnMen)
      (= ?map BeiJingMap)
    )
  )
)
)
```

Fig 4 A Tell and Ask-All advertisements

<p>Frame : Beijing Transportation</p> <p>Range: Beijing (Map)</p> <p>Purpose: Transport nivation</p> <p>Expected State: call the frame</p> <p>Member : {all the district in Beijing}</p>	<p>Frame: Expected state</p> <p>User Authority: Beijing govement</p> <p>State : good</p> <p>Evaluation: call frame Evaluation</p>	<p>Frame : User Authority</p> <p>User ID: city</p> <p>Password: ***</p> <p>Read Write: rw</p>	<p>Frame: Evaluation</p> <p>Method Name: Macro-Planing</p> <p>Address: http://evaluation.com:50</p> <p>Time : 1999/10/1</p>
--	---	---	---

Fig 5 The Conception of Beijing Transportation Environment

speed in the street, but traffic speed in the street is dynamically changed. As a result, you calculate the route with minimum time at t1, but at t1+Δt, the route cannot be the route with minimum time. It is a Conditional Forecast Problem.

Tagent_2 know the query is a route plan problem. First, it access the work flow from the knowledge base with the identifier task_2. It give the Tagent_2 instruction of how to work. He first abstracts the meta-node from the query, i.e.

meta-node1 = routes from A to B ;

meta-node2 = route with lest time from A to B

the 2 meta-nodes can consist of the query above. So, the following task is to find a partial-order chain in the sub-plan-node base where reusable sub-plan node is saved. The base also save the relationship between these nodes. The start point is meta-node1, and end-point is meta-node2. With one search method, we get one node chain:

T1 “solving accessible route set”,

T2 “choice the predictive point along one

accessible

T3 “predicting traffic speed with Datamining Agent at choiced sites”,

T4 “cooperation between different datamining in order to get more precise prediction value”,

T5 “solving the Minimum Time Route by analyzing the predicted speed value”,

T6 “ harmonize the all available route according to the state of Beijing transportation given, return optimal one”,

T7 “return the result to User”.

T1 and T2 is implemented in the Tagent_2. For there may be several routes available in T1, so T1 and T2 can be arried out asynchronously. T3 is implemented in DataMining Agent. It only knows the site for data mining, so T1, T2 and T3 can also be carried out asynchronously. T4 is in DataMining Agent. For the same reason, T1, T2, T3 and T4 can be carried out asynchronously. But some are carried out inside the same Agent, others inter-Agent. But, to T5, in theory, after getting all the

predictive speed value along the available route, it just begins to calculate. T5 is running in Tagent_2. After get the route with minimum time, send it to its User Agent. It is a loop from T1 to T5. With it, all the queries from users are met. But, as stated above, we must avoid the traffic congestion. So, T6 will harmonize the all available route. Because the transportation "State" is good, so T6 chooses corresponding evaluation method from <http://evaluation.com:50>. The result is that the traffic congestion is avoided as possible as could. At the same time, users also get satisfied route.

6. Conclusion

Our work just begins. There are lots of problems failing to be solved. Our future work will focus upon task plan between Agents and Data Mining based on Multi-Agent system.

Reference

- Qing Chen, Parvathi Chundi, 1998, Dynamic-Agents for Dynamic Service Provisioning, Software Technology Laboratory, HP Laboratories, IEEE.
- Peter C. Lockemann, Ulrike Kolsch, Arne Koschel, Ralf Kramer, Ralf Nikolai, Mechtild Wallrath, 1997, The Network as a Global Database : Challenges of Interoperability, Proactivity, Interactiveness, Legacy, Proceedings of the 23rd VLDB Conference Athens, Greece
- Tim Finin, Yannis Labrou and James Mayfield, 1995, KQML as an agent communication language, ffinin,jklabrou,mayfieldg@cs.umbc.edu
<http://www.agent.org>
<http://www.csee.umbc.edu/agents/>
<http://www.cs.umb.edu/kqml/>