# 2

# Cooperative Agent Learning Model in Multi-cluster Grid

Qingkui Chen, Songlin Zhuang and He Jia, XiaoDong Ding
*School of Computer Engineering,*
*University of Shanghai for Science and Technology*
*China*

## 1. Introduction

With the rapid development of the information techniques and Internet, and their popular application, the research results based on CSCW became the key techniques to build the enterprise information infrastructure (Raybourn & Newman, 2002).The computer based on cooperative work environments is playing the more and more important role in the business behavior of the enterprise today. Especially, the CSCW contains a lot of computation-intensive tasks and they need to be processed in some high performance computes. On the other hand, Intranet is more and more extended and a great number of cheap personal computers are distributed everywhere, but the utilization rate of their resources is very low. The researches of papers (Markatos & Dramitions, 1996) (Acharya&Setia, 1998) point out that many resources are idle in most network environments at a certain time. Even it is the busiest time of a day, still one third of their workstations aren't used completely. So, the paper (Chen & Na, 2006) proposed a framework which is how to collect and use the idle computing resources of CSCW environment that composed of multi-clusters connected by Intranet. It uses the idle computing resources in CSCW environment to construct a Visual Computational System (VCE). VCE can support two kinds of migration computations: (1) the Serial Task based on Migration (STM); (2) the Task based on Data Parallel Computation (TDPC). For adapting these heterogeneous and dynamic environments, we use the Grid (Foster & Kesselman, 1999) techniques and multi-agent (Osawa, 1993) (Wooldridge, 2002) techniques, and collect the idle resources of CSCW environment to construct multi-cluster grid (MCG). Because the migration of computing task and the dynamic changes of the idle resources in MCG, the intelligence of computing agents for raising the utilization rate of idle resource becomes very important. So, the effective learning model of agents is the key technique in multi-cluster grid. There are a lot of researches about agent learning (Kaelbling et al., 1996)( Sutton, 1988)( Watkins & Dayan,1992)( Rummery&Niranjan, 1994)( Horiuchi&Katai, 1999) nowadays. It includes two aspects: passive learning and active learning. The main theory of active learning is the reinforcement learning. At the same time, the agent organization learning (Zambonelli et al., 2000) (Zambonelli et al., 2001) model become another focus. The problem of distributed and cooperative multi-agent learning is studied through complex fuzzy theory in the paper (Berenji&Vengerov,

2000). But all these methods can't satisfy the need of dynamic multi-cluster grid. Especially, owing to the migration of the cooperative computing team, which is a group of cooperative computing agent to support data parallel computing, and the dynamic changes of grid idle resources, the cooperative learning model is very important for cooperative computing.

This chapter proposed a cooperative learning model of agents in multi-cluster grid that is composed of many computer-clusters connected by Intranet. By using the idle resource state of computer and the cooperative idle resource state, the state space is constructed; by using the sub-actions, the action space is built; through the state space and the action space, we construct dynamic rule space; through the reinforcement learning, the dynamic rule is revised. This model can support the grid computing and pervasive computing based on task-migratory mechanism, and it can fit the heterogeneous and dynamic network Environment. The experimental results show that this model can increase the percentage of the resources utilization in CSCW environment.

## 2. VCE

We collect and use the idle computing resources of CSCW environment to construct a Visual Computational System (VCE).There are two kinds of migration computations in VCE. The first is STM (Serial Task based on Migration). This kind of the task is the traditional serial task, and its description is based on the task packet. By delivering the task packet in CSCW environment, the task is executed by the computational service agents running on many idle computers in CSCW. The computational process is that the task packet is migrated for executing. The second is TDPC (the Task based on Data Parallel Computation). Making use of the overlapped size of segments of idle time of some computers in CSCW networks, we cluster these computers to become some Logical Computer Clusters (LCC), and each LCC is composed of the computers whose segments of the idle time are homology. So TDPC is the task running on LCC of CSCW environment.

In order to support the migration computations of VCE, a Grid Computation Environment was designed and implemented. It includes six parts and the architecture is presented as Figure 1.

**CSCW** is the traditional cooperative design system. Grid Computation Environment only concerns the Role Space (RS) and the Work Flow Space (WFS) of CSCW. **PCE** (Physical Computational Environment) is the physical computational environment of CSCW, and it is composed of many computer clusters connected by Internet or Intranet. PCE is the physical basis of CSCW, SDS, VCE and MAS. The single computational node (SC) and logical computer cluster (LCC) compose **VCE** which support STM and TDPC, and SC and LCC are called computational component (CC). The core of Grid Computation Environment is **MAS** (Multi-Agent System**)** which is composed of multi-Agents system.MAS can manage the Grid resources, do match works between the two kinds of the migration computations and computational component of VCE and execute computation. As we can see from Figure 1, MAS include four sub-systems, which are TAS (Task Agent System), SAS (Service Agent System), RAS (Resource Agent System) and MCAS (Matching Agent System). The four agent sub-systems of MAS communicate with each other by four shared data tables, and these tables are the task table (TT), the Service Table (ST), the Resource Management Table (RMT), and the Running Service Table (RST).TT is the set of all computing tasks whose state

is *"Committed"*. ST is the set of all computational services. RMT is the set of all computational components of VCE. RST is the set of all binding elements, and a binding element is a binding relation among the computing tasks, the computational services and the computational components.TAS receives the computing tasks from the users of RS and commits them to the table TT, and then monitors their states. When the computing tasks were "Completed", TAS delivers their results to the original user or WFS.SAS receives the computational services from the users of RS, then SAS checks up their legitimacy and register them to ST, and saves their service code to **SDS**. SDS can store the data of Grid Computation Environment and all computational classes. The MCAS, whose function is the match work between the migration computations and the computational components, runs on the Master of Grid Computation Environment.The main functions of RAS are the computational resources management, VCE monitor and the task scheduler. **Master** is a special computer, and it controls Grid Computation Environment.

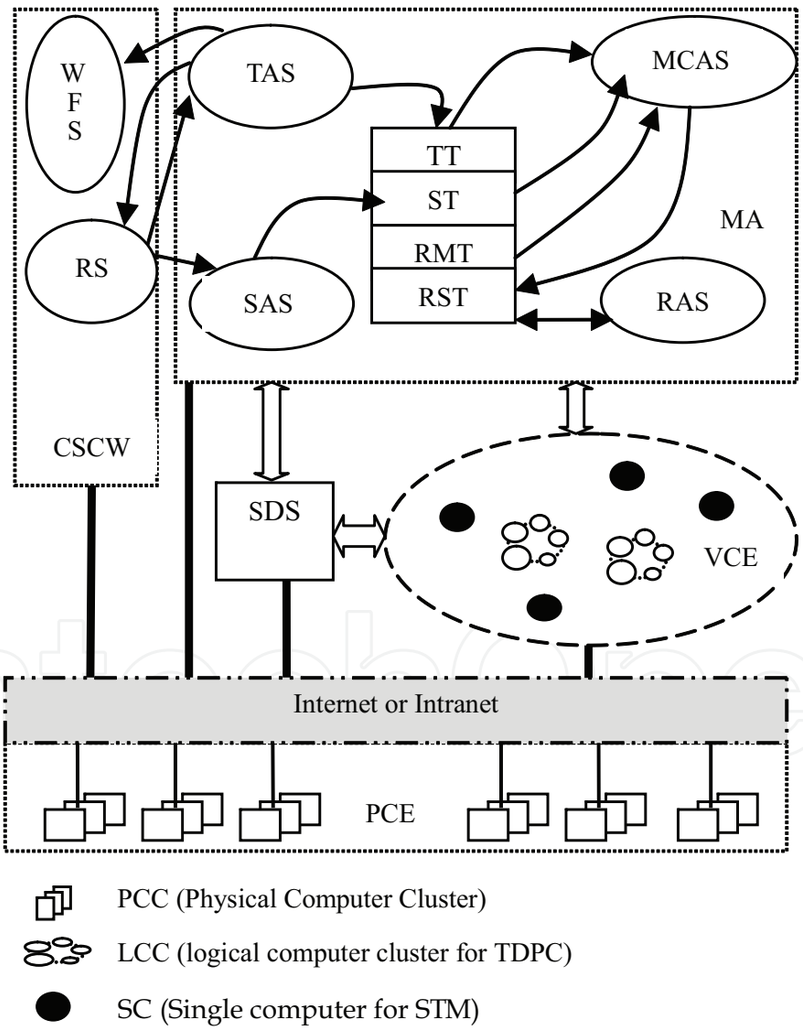



Fig. 1. Architecture of Grid Computation Environment

## 3. Architecture of MCG

In order to simplify the problem, we construct MCG (Multi-cluster grid) which is the simplification of Grid Computation Environment. MCG is composed of the idle computing resources of CSCW environment and support the computation-intensive tasks of CSCW. The Computation-Intensive Task needs very long time to run, and also needs high performance computer to support, such as the finite element analysis. For the description of learning model, we introduce some definitions:

**Computing Node (CN)** is defined as *CN* (*id*, *CT*, *Am*, *AS*), where *id* denotes the identifier of *CN*; *CT* denotes the type of computing node; *Am* denotes the main control agent of *CN*; *AS* is the set of agents running on *CN*.

**Computer cluster (CC)** is defined as *CC* (*Ma*, *CS*), where *Ma* denotes the main computer of *CC*; *CS*= {*CN1, CN2…CNp*} denotes the set of all computing nodes which *CC* includes;

**Computing Agent (CA)** is defined as *CA* (*id, PRG, BDI, KS, CE*), where *id* denotes the identifier of *CA*; *PRG* denotes the executable program set of *CA*; *BDI* is the description of its BDI; *KS* is its knowledge set; *CE* is its configuration environment.

*CA* is the basic element of executing computation task and support STM. If *CA* could complete the task independently, we call it the **independent computing agent** (*ICA*). If *CA* couldn't complete the task independently, and it must cooperate with others, we call it the **cooperative computing agent** (*CCA*).

**Cooperation Computing Team (CCT)** is defined as *CCT* (*id, Am, CAS, BDI, CKS, CCE*), where *id* denotes the identifier of *CCT*; *Am* denotes the main control agent of *CCT*; *CAS* denotes the set of all cooperative computing agents which *CCT* includes; *BDI* is the description of its BDI; *CKS* is its knowledge set. *CCE* is its configuration environment. *CCT* can support the TDPC in the logical computer cluster.

**Global Computing Group (GCG)** is defined as *GCG* (*id, Am, ICAS, CCTS, GKS, GCE*), where *id* denotes the identifier of *GCG*; *Am* denotes the main control agent of *GCG*; *ICAS* denotes the set of *ICA* which *GCG* includes; *CCTS* denotes the set of *CCT* which *GCG* includes; *GKS* is its knowledge set. *GCE* is its configuration environment.

**Multi-cluster grid (MCG)** is defined as *MCG* (*Ma*, *CCS*, *N*,*R, GCG*), where *Ma* denotes the main computer of *MCG*; *CCS* denotes the set of all computer clusters which *MCG* includes; *N* is the connection network set of *MCG*; *R* is the rules of connections; *GCG* is the global computing group.

Many tasks are executed together in *GCG* during the same time, and the tasks are calculated by a lot of *ICA*s or *CCT*s.

## 4. Rule descriptions

For the description of learning model, we introduce some definitions at first, and then described the organization method for the multi-agent rules.

**A basic Rule (*br*)** is defined as *br* (*id, rul, MRS*), where *id* denotes its identifier; *rul* denotes the formalization description of *br*; *MRS* denotes the meta-rule set for revising *br*.

**Basic Rule Set (BRS)** is the set of all the basic rules which *GCG* includes.

The state information in *MCG* can be decided by the scale of its idle computing resource, the computing task information, and the shared resource information. The state conception is described as follows:

**A state (*st*)** is defined as *st* (*id, sort, CSET*), where *id* denotes the state identifier; *sort* denotes the sort of this state; *CSET*= ($c_1$,$c_2$…$c_k$) denotes its condition vector. The number of the

condition elements which condition vector includes is k, and all states of the same sort have the same value of *k*. If the condition vector of a state only includes single computing node information, this state can be called *the independent state*; if the condition vector of a state includes the complex information about many computing nodes, shared network, etc., this state can be called *the cooperative state*. All states in *MCG* formed *the state space* (*SSP*).

**An action (Ac)** is defined as *ac* (*id,* μ, *AcDes, Revis*), where *id* denotes the action identifier; *AcDes* denotes the formalization description for this action. *Revis* is the set of revised strategies.

**A sub-Action** of *ac* can be defined as *ac* (*id,* μ, *AcDes*(μ), *Revis*), where *id* denotes the action identifier; μ is a decimal and 0<μ<1, *and* μ denotes the sub-action factor; *AcDes*(μ) denotes the formalization description for this sub-action. *Revis* is the set of revised strategies that make *AcDes* become stronger or weaker through the factor μ, and the revised result is *AcDes*(μ). So, a sub-action can be generated based on its father action *ac* and can be presented as *ac* (μ). All actions and their all sub-actions in *MCG* formed *the action space* (*ASP*).

**Dynamic Rule (*dr*)** is defined as *dr*(*id ,st, ac, br, w, sta, life*), where *id* denotes the rule identifier; $st \in SSP$, $ac \in ASP$, $br \in BRS$; *sta* is the state of *dr*, and $sta \in$ {"*Naive*", "*Trainable*", "*Stable*"}; "*Naive*" denotes that the *dr* is a new rule; "*Trainable*" denotes that the *dr* is revised rule; "*Stable*" denotes that the *dr* is a mature rule; *w* denotes the weight value of *dr*; *life* denotes the value of its life. The *dr* means that if the state is *st* then do sub-action *ac* (μ).If the state of *dr* is independent state, the *dr* is called *the independent rule*; if the state of *dr* is cooperative state, the *dr* is called *the cooperative rule*. The formed process of the dynamic rule and the relations between *st, ac* and *dr* are presented in Figure 2. All the rules in *MCG* formed *the rule space* (*RSP*). *RSP*, *ASP* and *SSP* formed GKS of *GCG*.

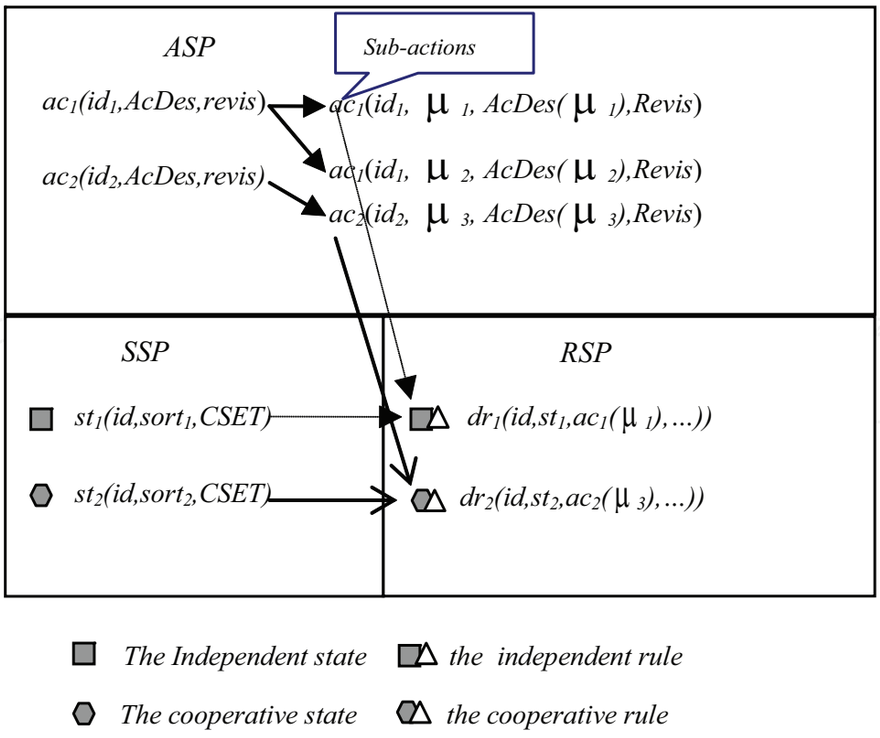



Fig. 2. The relations between *st*, *ac* and *dr*

If *dr* is a dynamic rule and *dr.w*>*MaxWeight*, and *MaxWeight* is a constant in *MCG,* we call *dr* the static rule (*sr*). Its state is "*Static*".
If *dr* is a dynamic rule and *dr.w*<*MinWeight*, and *MinWeight* is a constant in *MCG,* we call *dr* the castoff rule (*cr*). Its state is "*Castoff*".
The state graph of rules is presented in Figure 3.
The dynamic knowledge is the set of all the dynamic rules in *MCG*. The static knowledge is the set of all the static rules in *MCG*. The basic knowledge can be formed by passive learning. For adjusting the dynamic knowledge established by the basic knowledge, we can revise them through the reinforcement learning and the multi-agent cooperation during the computing process.
Task (TSK) *is* defined as *TSK* (*id, DAT, AS*), where *id* denotes the identifier of *TSK*; *DAT* denotes the data set; *AS* is the set of agents which execute *TSK*.

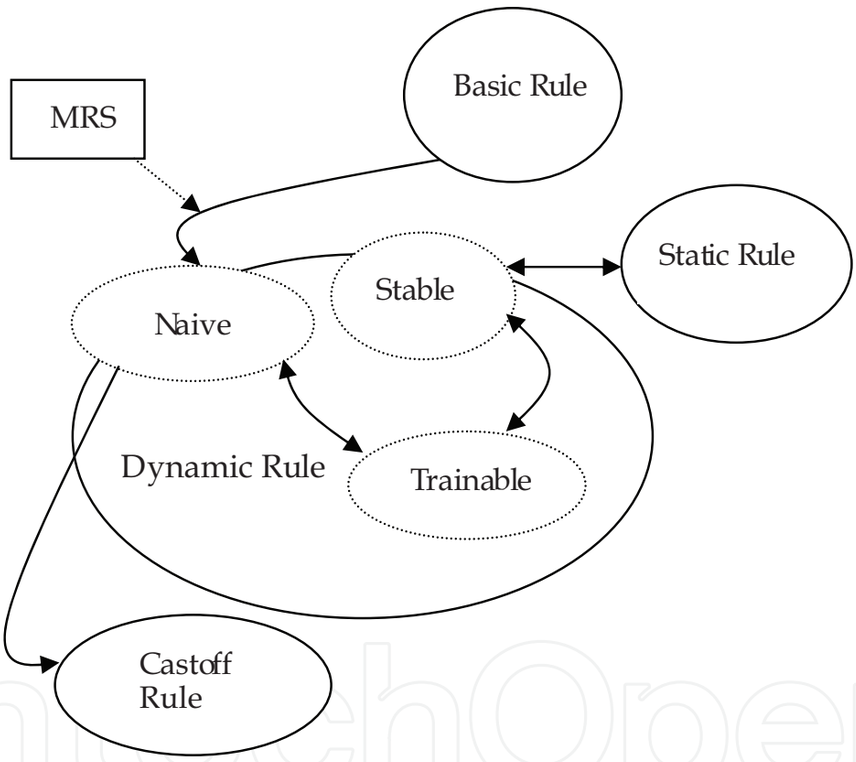



Fig. 3. State graph of rules

## 5. Description of learning model

### 5.1 Initialization of knowledge

**Algorithm1 (Initialization of basic rules)**
(1)The users construct the *BRS* through *MRS* and initialize the state space *SSP*;
(2)Construct the initialization action space *ASP*:
Decide the action set *TMP* by users;
Set *ASP*=φ;
While *TMP*≠φ Do

Get an action *ac* from *TMP*;
Set $\mu$=1;
Construct the *AcDes* and *Revis* for *ac* by users;
Set *AcDes(μ) =AcDes*;
Construct the sub-action *ac (μ);*
ASP=ASP+ {ac (μ)};
TMP=TMP-{ac};
End while;

(3)Construct the initialization rule space *RSP*:
*TMP=SSP*;
*RSP*=φ;
While *TMP*≠φ Do
Get a state *st* from *TMP*;
Choose a sub-action *ac* (μ) from *RSP* through the *st.sort* by users;
Construct a new dynamic rule *dr*:
{*dr.st=st*;
*dr.ac=ac (μ)*;
*dr.w*=0;
*dr.sta=“Naive”*;
*dr.life*=0;
}
Add the *dr* to *RSP*;
TMP=TMP-{*st*};
End while;

(4)Output *RSP*, *ASP* and *SSP*;
(5)End.

For fitting the variety of computing device resources, the dynamic rules must be generated and the process is described as follows:

**Algorithm2 (Generation of dynamic rules)**
Input: a new state *st* (*id, sort, CSET*), *RSP*, *ASP* and *SSP*;
Output: the new *RSP, SSP* and *ASP*;
(1)Get the sort of *st* and save it to *TmpSort*;
(2)Get all states with the same sort of *TmpSort* from *SSP*, and form the state set *SubSSP*;
(3)Construct condition relation matrix:
Suppose that *TmpSort* sort state has *k* conditions, and the number of the elements in *SubSSP* is *m*; these condition vectors are ($c_{11}$, $c_{12}$…$c_{1k}$), ($c_{21}$,$c_{22}$…$c_{2k}$)… ($c_{m1}$,$c_{m2}$…$c_{mk}$) and the condition vector of *st* is ($c_{m+1,1}$,$c_{m+1,2}$…$c_{m+1,k}$); All $c_{ij}$($1\le i\le m+1$, $1\le j\le k$) are integer and can be defined by real applications;
Construct a (*m*+1) x*k* matrix *CM*= {$c_{ij}$| ($1\le i\le m+1$, $1\le j\le k$)} by using all condition vectors of states in *SubSSP*;
(4)Construct the fuzzy matrix *FM*= {$f_{ij}$| ($1\le i\le m+1$, $1\le j\le m+1$)}, where
$f_{ii}$=1;
$f_{ij}=1-\beta\left(w_1|r_{i1}-r_{j1}|+w_2|r_{i2}-r_{j2}|+\ldots+w_k|r_{ik}-r_{jk}|\right)$,
when $i\neq j$, $0<\beta<1$, $w_1+w_2+\ldots+w_k=1$, and $w_l>0$ ($1\le l\le k$);
(5)Construct the fuzzy equivalence matrix (please refer the document (Zadeh, 1975)):

Repeat do
*FT=FM* ⊙ *FM*; // ⊙ is the operation theorem to take the maximum and minimum.
If *FT=FM*, then go to (4);
*FM=FT*;
End repeat;
(6)Calculate the *th*-cut matrix *FMth* according to the threshold *th* , and *th* ∈ [0, *Z1*] or *th* ∈ [*Z1*, *Z2*] or *th* ∈ [*Z2*, *Z3*] or *th* ∈ [*Z3*,1],0≤*Z1*≤*Z2*≤*Z3*≤1,where *Z3* is the minimum of the *effective threshold*, *Z2* is the minimum of the *average threshold*, *Z1* is the minimum of the *artificial threshold*; (Refer to step (9))
(7)Divide the *FM* into several equivalent classes *SFM1* ⋃ *SFM2* ⋃ …*SFMe* according to *FMth*; The element number of every equivalent class must be greater than 1, and it can decrease the value of *th*;
(8)Choose the equivalent class *SFM* that includes the condition of *st*;
Form the candidate rule set *CRSET* from *RSP* through *SFM;*
(9)/*effective threshold*/
{
If *th* ∈ [*Z3*,1], then
{Get the dynamic rule *dr* which has the maximum weight in *CRSET*;
Construct a new dynamic rule *ndr* through *st* and *dr*;
*RSP=RSP*+ {*ndr*};
*SSP=SSP*+ {*st*}
};
*/*average threshold*/*
If *th* ∈ [*Z2*, *Z3*), then
{Calculate the *avm*=average (*dr.μ* | *dr* ∈ *CRSET*);
Search a dynamic rule *sdr* which satisfies |*avm-sdr.μ* | ≤ε (*sdr* ∈ *CRSET and* 0≤ε≤1);
If *sdr* existed, then
Construct a new dynamic rule *ndr* according to *st* and *sdr*;
Else
{Build a new sub-action *nac(μ)*according to *ac*, *avm* and *Revis*, Where *μ=avm*;
Construct a new dynamic rule *ndr* according to *st* and *nac(μ)*;
}
*RSP=RSP*+ {*ndr*};
*SSP=SSP*+ {*st*};
*ASP=ASP*+ {*nac (μ)*}
};
*/*artificial threshold*/*
If *th* ∈ [*Z1*, *Z2*) or *th* ∈ [0, *Z1*), then
{Calculate the *avm*=average (*dr.μ* | *dr* ∈ *CRSET*);
Build a new sub-action *nac(μ)* through *ac* and *avm* by *users*, Where *μ=avm*;
Construct a new dynamic rule *ndr* through *st* and *nac(μ)*;
};
*RSP=RSP*+ {*ndr*};
*SSP=SSP*+ {*st*};
*ASP=ASP*+ {*nac(μ)*};
}.

### 5.2 Life and weight of dynamic rule

According to the algorithm2, the dynamic rule is constructed, and it must be adjusted when the *TSK* is calculated. The adjusting mechanism can adopt the reinforcement learning, and the resources utilization rate of *TSK* can be considered as the reinforcement function.

**Algorithm3 (Adjusting weight and life for rules)**

Suppose that *Y1* is the *castoff threshold*, and *Y2* is the *mature threshold*; *Q(urt)* is the *reinforcement function* and *Q(urt)*>0, and *urt* is the resources utilization rate; *MaxWeight* is the maximum of the rule weight, and *MinWeight* is the minimum of the rule weight, and let *MinWeight*<*Y1*<*Y2*<*MaxWeight*; *MaxLife* is the maximum of life value.

(1)Suppose that a computing agent *CA* adopted a dynamic rule *dr* of *CA.KS*;
*dr.life*+ +; /* increase the value of life */
Wait for the *urt* from the computing system;
(2)If *urt*>0, then
*dr.w*=*dr.w*+ *Q(urt)*; /*increase weight*/
If *urt*<0, then
*dr.w*=*dr.w*-*Q(urt)*; /*decrease weight*/
(3)If *dr.w*>*MaxWeight*, then
*dr.w*=*MaxWeight*;
If *dr.w*<*MinWeight,* then
*dr.w*=*MinWeight*;
(4)If *dr.w*<$Y_1$ and dr.life>MaxLife, then
*dr.sta*= "*Castoff*"; /*Castoff rule */
(5)If $Y_2$<*dr.w*<MaxWeight, then
*dr.sta*="*Stable*"; /*Stable rule */
(6)If *dr.w*>= MaxWeight, then
*dr.sta*= "*Static*"; /*Static rule */
(7)If $Y_1$<*dr.w*<$Y_2$, then
*dr.sta*= "*Trainable*"; /*Trainable rule */
(8)If MinWeight<dr.w<$Y_1$, then
*dr.sta*="*Naive*"; /*Naive rule */
(9)End.

When the learning and executing process runs for a long time, the dynamic knowledge is excessive and it will reduce the searching efficiency. So, it needs the artificial adjustment for dynamic knowledge to reduce the number of the rules. The process is omitted.

The dynamic cooperation knowledge of *CCT* usually uses shared resources, such as the network bandwidth, so the value of *urt* is the average resources utilization rate of *CCT*.

### 5.3 Learning in migration process

In order to persist and improve the global knowledge in *MCG* , the main control agents *Am* in *CN*, *CCT* and *GCG* get the shared organization knowledge through the organization learning (Zambonelli et al., 2000)(Zambonelli et al., 2001). Owing to the agent migration in *MCG*, the running environments of *ICA* and *CCT* are in the dynamic change. For fitting the change, the migration learning of the organization is very important.

**Algorithm4 (Migration learning of *ICA*)**

An *ICA* running on the computing node *CNi* must be migrated to the computing node *CNj*, so the migration learning process is presented as follows:

(1)*ICA* commits its *KS* to *Am* in *CNi,* and *Am* saves the *KS*;
(2)*ICA* asks for a new computing node *CNj* from *MCG* and consults with *Am* of *CNj*, and migrates into *CNj*;
(3)*ICA* gets the current state set *CSS* in *CNj* and searches the corresponding dynamic rule form *RSP* according to *CSS*, then refreshes it's *KS*:

*TMP*=*CSS*;
*OTHER*=φ;
While *TMP*≠φ Do
Get a state *st* from *TMP*;
Search a dynamic rule *dr* from *RSP* by *st*;
If *dr*≠φ, then /* successful*/
Add *dr* to *ICA.KS*;
*TMP*=*TMP*-{*st*};
Else
*TMP*=*TMP*-{*st*};
*OTHER*=*OTHER*+{*st*};
End if;
End while

(4)If *OTHER*≠φ, then
While *OTHER*≠φ Do
Get a state *st* from *OTHER;*
Start the algorithm2 to generate a new dynamic rule *ndr* above *st*;
Add *ndr* to *ICA.KS*;
*OTHER*=*OTHER*-{*st*};
End while;
End if;
(5)*CNi* and *CNj* report their *KS* and *CE* to *RSP, ASP and SSP* in *GCG*;
(6)*ICA* continues executing and learning on *CNj*;

**Algorithm5 (Migration learning of CCT)**
Suppose that *CCT*(*id, Am, CAS, BDI, CKS, CCE*) is a cooperation computing team running on the computer cluster *CCi*, *CAS*=*MIG*⋃*NMIG*, where *MIG* is the set of the computing agents that will be migrated; *NIMG* is the set of the non-migration computing agents. The migration learning process is as follows:
(1)All the computing agents of *MIG* migrate to their new computing nodes respectively and learn the new knowledge according to the algorithm4;
(2)All *CA* ∈ *MIG*⋃*NMIG* commit their state information to *Am*; *Am* calculates the new cooperative state information set *CISET*;
(3)If *CISET*≠φ then
*Am* generates a new dynamic set *NRSET* according to *CISET*, and the process is similar to the step (3) (4) in aglorithm4.
*Am* refreshes *CCT.CKS*;
End if;
(4)*CCT* start the calculation work in a new environment;
(5)End.

### 5.4 Learning process of GCG

**Algorithm6 (GCG learning process)**
(1)*GCG* calls the algorithm1 to do initialization work;
(2)*GCG* receives a task *TSK*;
(3)*GCG* allots a computing device *PE* (that is a *CN* or a *CC*) for *TSK*, and starts the computing agents for *TSK*;
(4)While *TSK* is not finished Do
All computing agents cooperative calculate the *TSK*;
If *TSK* must be migrated, then
Start the migration learning algorithm (4 and 5);
Calculate the resource utilization rate *urt*;
Start the algorithm3 to adjust the weight and life of dynamic rules;
Refresh ASP, SSP, and RSP;
End if;
End while;
(5)End.

## 6. Experiments

We construct a *MCG* that is composed of 12 computers and 3 computer-clusters that connected by Intranet. The independent state of the computer can be decided by its idle CPU resource, idle memory resource, idle disk resource and idle net adapter resource, so, the condition vector of the state is (*CPU*, *memory*, *disk*, *net adapter*). The cooperative state of *CCT* can be decided by the *CCT* scale, the bandwidth of *CCT*, the total idle CPU of *CCT*, the total idle memory, the total idle disk of *CCT* and the total net adapter power. The *ASP* is the strategy set, such as the buffer size, the size of basic communication unit and the synchronous time, etc. The computing tasks provided by *MCG* are the matrix operations and the linear programming. The *CCT* algorithms (Parallel algorithms based on computer cluster) for the matrix operations and the linear programming are given. The Intranet clock is synchronous by GTS protocol. The initial basic rules include 24 rules for *ICA* and 7 rules for *CCT* and the parameter values are as follows: *MaxLife*=43200(s), *Y1*=15, *Y2*=80, *Y3*=0.3 ,*Z1* =0, *Z2*=0.6,*Z3*=0.9, *MaxWeight*=100 and *MinWeight*=0. The experiment includes seven times and each time has 12 hours, and the total time is 84 hours. The tests adopt a random function to choose some tasks (the matrix operation, the linear programming and their parallel edition) in each time. In order to make the tasks migrate as far as possible in the *MCG*, We make use of the random migration function *RandMigration()* and form the migration strategy during the test processes. Through the average values of the test information, we observe the learning results of this model. The experiment results are as follows:
The changes of the average resource utilization rate of *MCG* along with the learning process have been tested. The thick solid line means the utilization rate distribution in the figure 4(a). This test result shows that this model can raise the resource utilization rate of *MCG*.
The changes of the number of new dynamic rules which are generated during the learning process have been tested. The solid line means the distribution of the dynamic rules whose state is always the "*Naive*" during their life period in the figure 4(a). The dotted line means

the distribution of the *"Trainable"* dynamic rules whose state has become the *"Stable"* during their life period. The test results show that the learning efficiency of this model increases gradually along with the learning process.

The number of the artificial adjustment has been counted. The solid line means the distribution of the number during the tests in the figure 4(b). This test result shows that the number of artificial adjustments decreases gradually along with learning process.

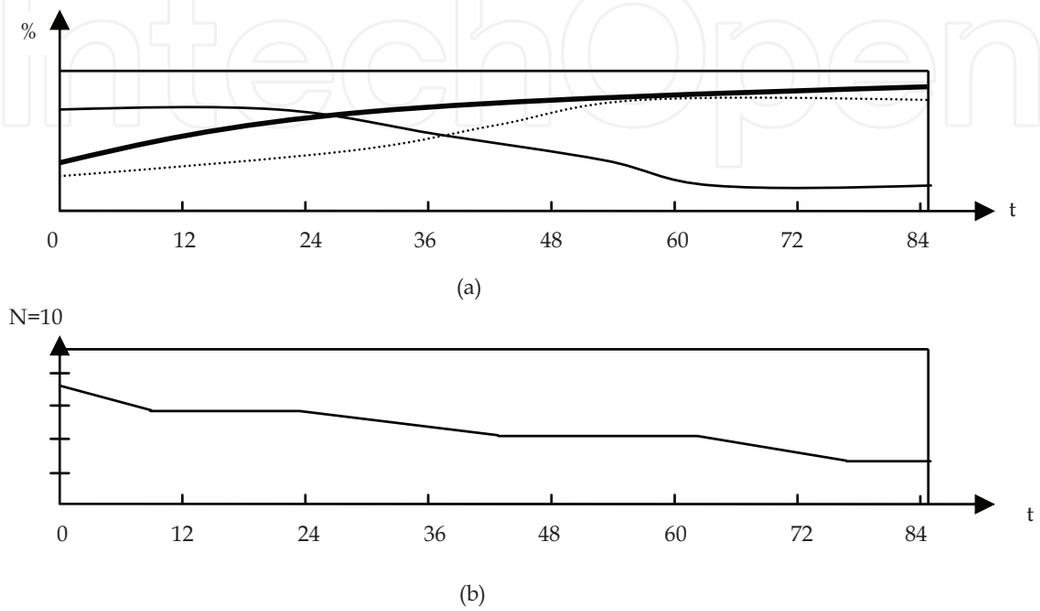



Fig. 4. The results of tests

## 7. Conclusions

In order to support the computation-intensive tasks, we collect the idle computational resources of CSCW environment to construct the multi-cluster grid. Because of the heterogeneous resources, the state of the idle computing resources changes in the process of the computing and the task migration. For fitting the state changes, the dynamic rule mechanisms of agents are proposed. According to the Grid techniques, Computing Agent, Cooperation Computing Team, the state space, the action space, the dynamic rule space, the reinforcement learning and so on, a cooperative learning model of agent was designed and implemented in this chapter. In the multi-cluster environment, using resources effectively is very difficult for the grid computing, however, a good learning model can improve the intelligence of multi-agent and also can raise the resources utilization rate. We do the experiment and the results prove that the cooperative learning model of agent which is introduced in this chapter can not only improve the intelligence of multi-agent, but also increase the utilization rate of idle resources in CSCW.

## 8. References

Acharya,A.&Setia,S.(1998).UsingIdle Memory for Data-Intensive Computations, *Proceedings of the 1998 ACM SIGMETRICS Conference on Measurement and Modeling of Computer*

*Systems*, pp.278-279, ISBN :0163-5999,Madison, Wisconsin, USA, 1998.6, ACM New York, NY, USA.

Berenji,H.R.&Vengerov,D.(2000). Advantages of Cooperation Between Reinforcement Learning Agents in Difficult Stochastic Problems. *Proceedings of theNinth IEEE IntConf on Fuzzy System*, pp. 871-876, ISBN: 0-7803-5877-5, San Antonio, TX, 2000.5, IEEE press.

Chen, Q.K.&Na,L.C.(2006).Grid Cooperation Computational System for CSCW, *Proceedings of the 10th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD2006)* , ISBN: 1-4244-0963-2, Nanjing, China,2006.5, IEEE Press, New York,USA.

Foster,I.&Kesselman,C.(1999).The Grid: Blueprint for a New Future Computing Infrastructure, Morgan Kaufmann Publishers, ISBN: 1558609334,San Francisco, USA.

Horiuchi,T.&Katai, O. (1999).Q-PSP learning: An exploitation-oriented Q-learning algorithm and its applications, Transactions of the Society of Instrument and Control Engineer, VOL.35, NO.5 (1999), pp.645-653, ISSN: 0453-4654.

Kaelbling, L.P.; Littman, M.L. &Moore, A.W.(1996). Reinforcement learning: A survey, Journal of Artificial Intelligence Research, Volume 4,pp.237-285, ISSN: 1076-9757.

Markatos,E.P.&Dramitions, G. (1996). Implementation of Reliable Remote Memory Pager, *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, pp.177-190 , San Diego, CA, 1996.1, USENIX Association , Berkeley, CA, USA.

Osawa,E.(1993).A Scheme for Agent Collaboration in Open MultiAgentEnvironment .*Proceedings of the 13th international joint conference on Artificial intelligence - Volume 1*,pp.352-358, Chambéry, France,1993.8, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Rummery, G. Niranjan, M.(1994).On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Raybourn, E.M.& Newman, J. (2002).WETICE 2002 Evaluating Collaborative Enterprises Workshop Report, *Proceeding of the Eleventh IEEE international Workshops on enabling technologies: Infrastructure for Collaborative Enterprises*, pp.11-17 , ISBN : 0-7695-1748-X, Pittsburgh, Pennsylvania, USA, 2002.6, IEEE Computer Society ,Washington, DC, USA.

Sutton,R.S. (1988).Learning to predict by the methods of temporal differences, Machine Learning, pp.9-44, ISSN: 0885-6125 (Print) 1573-0565 (Online).

Watkins, C.J.C.H. & Dayan, P.(1992).Q-learning, Machine Learning, pp.279-292, ISSN: 0885-6125 (Print) 1573-0565 (Online).

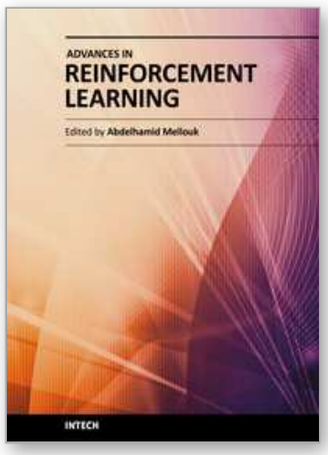
Wooldridge, M.(2002).An Introduction to MultiAgent Systems, John Wiley & Sons (Chichester, England). ISBN: 0 47149691X.

Zadeh,L.A. (1975).The concept of a linguistic variable and its application to approximate reason-ing, *Information Sciences*, Vol. 8, No. 3. (1975), pp. 199-249.

Zambonelli, F.; Jennings, N.R. &Wooldridge, M. (2000).Organizational abstractions for the analysis and design of multi-agent systems, *Proceedings of the1st International*

*Workshop on Agent-Oriented Software Engineering*, pp.127-141, ISBN: 3-540-41594-7, Limerick, Ireland, 2000, Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Zambonelli, F.; Jennings, N.R.&Wooldridge,M.(2001).Organizational rules as an abstraction for the analysis and design of multi-agent systems. International Journal of Software Engineering and Knowledge Engineering, pp.303-328, ISSN: 0302-9743 (Print) 1611-3349 (Online)




**Advances in Reinforcement Learning**
Edited by Prof. Abdelhamid Mellouk

ISBN 978-953-307-369-9
Hard cover, 470 pages
**Publisher** InTech
**Published online** 14, January, 2011
**Published in print edition** January, 2011

Reinforcement Learning (RL) is a very dynamic area in terms of theory and application. This book brings together many different aspects of the current research on several fields associated to RL which has been growing rapidly, producing a wide variety of learning algorithms for different applications. Based on 24 Chapters, it covers a very broad variety of topics in RL and their application in autonomous systems. A set of chapters in this book provide a general overview of RL while other chapters focus mostly on the applications of RL paradigms: Game Theory, Multi-Agent Theory, Robotic, Networking Technologies, Vehicular Navigation, Medicine and Industrial Logistic.



INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen