We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Research on Multi-Robot Architecture and Decision-making Model

Li Shuqin[1] and Yuan Xiaohua[2]
Department of Computer Science,
[1]Beijing Information Science &Technology University, Beijing
[2]College of Information, Shanghai Ocean University, Shanghai
China

1. Introduction

Robot architecture (or controlling architecture) mainly describes robot combining modules, then relationship between modules and exchanging between robots. Up to now multi-robot architecture is one of the main topics in multi-agent and multi-robot research, and many researchers strive to design controlling architectures of excellent performance, and a few of them have already proposed some valid multi-robot architecture and given related simulation [Dias 2004], among which the famous are

- Architecture of GOPHER [Caloud et al. 1990] combined by four layers including decomposition and distribution of task, moving programming and execution control. In GOPHER a central task processing system (CTPS) take charge of task distribution.Every robot can learn its task from CTPS's announcment, use task distributing algorithm to determine its own role, and use a classical AI programming technique to realize its role. Although has successfully fulfilled tasks of box pushing and tracing, GOPHER prevent robot join other task before fulfilled its current task, and has not clearly proposed how robot can restore from error or failing, and how to make use of the limited sources and to define state role. These limitations have weakened robot in GOPHER to work well under dynamic environment.
- Distributive architecture of ALLIANCE [Parker 1998] was a behavior-based, fault-tolerant and self-adoptive multi-robot cooperative architecture. In ALLIANCE, individual robot used a behavior-based controller to select behavior based on a motivation model. Robot in ALLIANCE could not make fast and optimal response in dynamic environment. ALLIANCE has not considered optimal distribution of limited resources, did not allow dynamic assigning of new type tasks too. By far ALLIANCE has fulfilled tasks such as box pushing, disc collecting, and formation moving.
- Lueth and Laengle [Laengle et al. 1998] co-proposed a distributive controlling architecture of KAMARA team oriented to multi-robot cooperation. KAMARA is behavior fault-tolerance and error correction also. The architecture is based on universal concept of agent to respresent robot component. Agent take charge of communication, task programming and behavior selecting, and task executing. In KAMARA, agent without capability can not take part in consultation and being assigned any task.

KAMARA can not guarantee incapable agent fulfilling its task. And in addition, in KAMARA there is no optimal resource exploit also, so agent need to store all the resources, thus will lead to a calamitous increase of consultation.

- The famous controlling architecture STEAM [Tambe & Zhang 1997] was built on joint-intension and sharing programming. STEAM does not rely on special domain knowledge, thus is reusability. In order to reduce communication, based on decision-making theory, STEAM used a communication selecting mechanism to guarantee the realization of joint-intension although without any communication. Tambe and etc. have further designed a consultation-based model of CONSA based on STEAM.

- Noreils[Noreils 1993]proposed a three-layer hierarchical controlling architecture , in which programming layer divides task into little sub tasks and assigns sub tasks to a robot network, controlling layer organizes robot in task fulfilling, and function layer provides actually controlling. Noreils has report implement of this architecture in multi-robot cooperation of box pushing.

- Habib [Habib et al. 1992] proposed an AC-tRESS controlling architecture, in which a consultation mechanism allow robot to seek help from other robots when needed.

- Zhao,Y W. & Tan, D L [Zhao & Tan 1990] proposed a hybrid hierarchical architecture based on behavior decomposition.

- Tan-Min and etc. [Tan et al. 2005 ;Chao et al.2001] proposed a hybrid controlling architecture oriented to task-level cooperation of multi-robot system, which was combined by layers of system monitor, cooperative programmer, and behavior controlling.

Since hierarchical cooperation can reduce programming complication and improve system efficiency, the above architectures are almost hierarchical, but these architectures have not emphasized autonomical behavior decision-making.

We take for that in multi-robot system under highly dynamic environment, it is impossible and unpractical to rely on one controller to assign task and to make route programming for all robots, robot must has capability of autonomous, self-adaption, and cooperation, all the three are of the same importance and can not be lacked.

In this chapter, in order to emphasize the autonomical behavior decision-making, and for system modularity and robusticity, we propose a hybrid architecture based on five layers, among which decision-making is explicitly being presented as one laye. Since in our architecture, different layer send out different information, communication consumptioncan is largely reduced, and when exigency occurrs, robot needs not to wait for induction from high layer, thus the system is more effective and robustic.

Firstly we introduce the hybrid architecture in section 2. Then we design the decision-making module and develop a related algorithm in section 3. In section 4 we gave details on implement of our architecture in Garbage disposal under dangerous environment, and at last in section 5, we arrived at some chapter conclusions.

2. Hierarchical robot architecture based on behavior

Based on the advantage of existing robot architecture, considering characteristics of moving task of multi-robot in dynamic complex environments, emphasizing robot's capability of self-adaptive, autonomous decision-making and cooperation, and strengthening monitoring of robot also, we proposed a behavior–based five-layer hybrid architecture of hierarchical individual robot, which includes two modules and five layers as shown in figure 1.
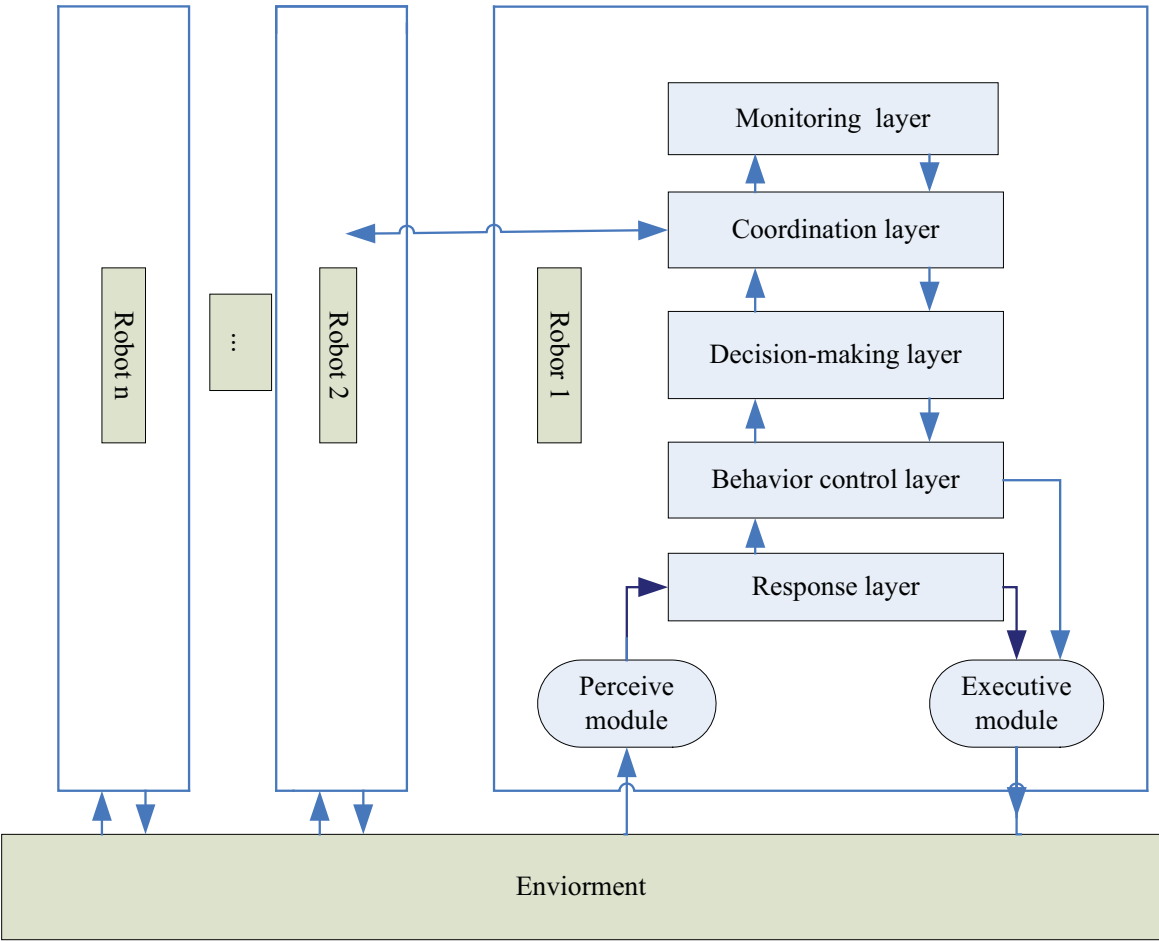
Fig. 1. Five-layer hybrid hierarchical robot architecture

Where the five layers are

1. **System Monitoring Layer**

Combined by monitoring modules, this layer watch if system state is abnormal. If it is, then halt task execution and send a message to behavior planning controller.

2. **Organization and coordination layer**

This layer mainly take charge of task management and make robot cooperation rules. It consists of seven modules as below

- Repository stores knowledge of environment, task, robot and experience. Robot knowledge are about other surrounding robot, especially robot's location, pose, and intension. Experiential knowledge are a set of examples, which robot canuse for reference when organizing, coordinating and programming.
- Update module periodically modify and update its interior organizing and coordinating database according to information it collected.
- Task decomposition module will partition system functions according to knowledge of tasks, and to build up robot organization model.
- Task assessment module will assesses the cost and befit of task fulfilling.
- Task assigning module. According to different robot role, this module assigns tasks using a cost contract network or of optimization.
- Communication module mainly transfers local layer information among robots.

- Cooperation module is the kernel controller in the layer. By exchanging with the same module of other robot,it sets up a hierarchical organization, and by cooperatively assessment, decomposition and assigning of task, it can fulfill task decomposition and assigning the subtasks quickly and rationally.

3. **Behavior decision-making layer**

This layer reflects robot autonomy and is mainly combined by behavior decision-making module and communication module. The first behavior decision-making module involves robot intention and realizes robot's cooperation intension. According to repository content and information from exchanging, this module autonomous search, reason and decide, produce joint behavior intension, that is, to choose a team to take part in. The second communication module mainly transfers information about behavior intension among robots.

4. **Behavior control layer**

This layer mainly takes charge of detail planning and executing of behavior decisions made by the above behavior decision-making layer. Behavior control layer is mainly combined by module of programming, coordination and communication. Programming module produce or select a recently behavior sequence according to tasks selected by behavior decision-making layer, when error or unexpected events occurring, programming module need to do reprogramming. Strategy coordination module coordinates robot moving to avoid collision and dead-lock as much as possible. Communication module transfers information about behavior programming and coordination.

5. **Response layer**

This layer is combined by response modules, and primarily give rapid response. In this layer there is a rule base which maps perceived information to some special behaviors. Behaviors of this layer have the highest priority. Two modules within response layer are

- **Perceiving module**

This module perceives and abstract environmental changes during robot moving Different kinds of information after abstracted will be loaded and processed in a related processing layer. For example if perceived some emergent, then information will be sent to response layer, from whose's response, behavior instruction will be sent to an Execute Module.

- **Execute module**

This module takes charge of actual behavior execution, such as avoiding obstacles, forward and backward movement, and etc.

In our architecture, time cost in decision-making layer is the longest. As refer to literature [Farinelli et al. 2003], time cost in response layer was about 10 ms, and that of higher layer was longer than 1min. In general, our proposed architecture has properties such as

- By setting a behavior decision-making layer to emphasize robot autonomous ability;
- Different layer will send out different information, thus can largely reduce communication consumption.
- When exigency occurring, robot can make response by itself, no need to wait decision from some higher layer, and
- System has modularity and robusticity.

3. Implement of behavior decision-making module

Robot can be taken as agent that has limited range of vision and communication, and certain autonomy. So robot in system has capabilities such as [Tang 2002, Xu 2004].

- Perceptive capability,it can perceive and adopt to the changes of environment.

- Communication capability, which is needed when robots consult for cooperation.
- Moving capability, which includes steering and moving in dynamic environment, such as climbing, cross country, paddle and etc. Moving capability can be represented by moving speed and direction.
- Behavior capability, which includes skills needed in task fulfilling, such as installing, maintaining, conveying, digging, site leveling out, attacking, scouting, computation, searching, and plotting and etc.
- Behavior decision-making capability, which is robot autonomy in behavior. In this chapter, this mainly refers that under multiple task condition, robot can decide to select a task team to join in according to information it collected.
- Real time response capability, which means that robot can response and take behaivior in emergency, such as avoiding obstacle and collision.
- Cooperation capability. In multi-robot system, robot must have capability of cooperation with others for task fulfilling.
- Capability of local programming, which means that robot can deduce other robot's intension, and according to this to plan behavior of its own.
- Organization capability, it refers that Leader robot can organize and coordinate robots with different skills to fulfill tasks together. And
- Learn capability. Individual robot must can learn from and adopt to complicate and changeful environment, thus to improve the running efficiency of the whole system.

The below studies are mainly related to robot decision-making module in our architecture. We firstly give definitions of robot intension, intension rule and role, then analyze factors influence robot behavior decision-making, at last propose a intension decision-making algorithm based on multiple dimension attributes.

3.1 Definition of intension

In dynamic environment, if can not fulfill some complicate task independently for lack of essential global environmental information or skills by itself, individual robot must has capability to choose a team to take part in, synthetically according to the environmental information collected by its perceptive system, intension of other robot within its communication range, and system runner's indirect instruction (called as joint intension), and through behavior programming to produce a series behavior to fulfill the intension .

Definition 1 Robot intension

Robot intension expresses robot's task selection at some time abiding by the intension rule defined as below.

Definition 2 Robot intension rule

There are two robot intension rules

- At a certain time intension only can be one, and
- intension must be of some certain stability and flexibility.

Here stability refers that intension can not change frequently, and flexibility is that intension can change in some special situation, such as robot encounter a new target, or intended task has being fulfilled, or motivation of one target disappeared. When accident comes forth in task fulfilling, individual robot should modify its intension also.

3.2 Definition of role

Because of robot's limited vision and communication capability, so in order to reduce communication, according to robot's knowledge of current task and the distance from the task, we give the definition of state role that can be referred to [Chaimowicz et al. 2002].

Definition 3 State role

In task fulfilling, the distance from robot to task and robot's responsibility are called state role, it is related to task in specified application. State roles can be converted dynamically. In our system, we set five state roles, including Explore, Leader, Approach, Attach and Arrived.

Definition 4 Leader

Leader is the first robot that arrives at task, and take charge of organizing of task fulfilling. To a given task, Leader should be the only. Represented by R_0, Leader can be described as

$Find(first, goal\_T) \wedge Distance(R_0, goal\_T) \le \rho$, where $\rho > 0$ is the nearest distance constant , and $goal\_T$ is goal of the task.

Definition 5 Arrived

Arrived is robot that had arrived at task location and waiting for fulfilling it, which can be described as

$Distance(R_0, goal\_T) \le \rho + \varepsilon$, where $\varepsilon > 0$ is a small positive constant. In our below experiments, $\rho + \varepsilon$ is equal to robot vision radius.

Definition 6 Attach

Attach is the robot within R_0's communication range or R_0 can indirectly communicate with it, and it has already selected some task and already begin to move to. Attach can be described as $Distance(R, R_0) \le R_0.Cap.Crange$.

Definition 7 Approach

Approach is also within R_0's communication range or R_0 can indirectly communicate with, Although Approach has got some task information, but it has not selected any task yet.

Definition 8 Explore

Explore are robot that move randomly in environment for searching task.

In system running, each robot will correspond to and execute one role at one moment. In dynamical environment robot's role can be reassigned, which include methods of Dynamical Assigning and Relocating. In Dynamical Assigning, robot can be assigned to a new role after it fulfilled a task. After all the tasks of one team having being fulfilled, the team will be dissolved, and Leader and Arriveds will be reassigned as Explore. When Leader is not competent, its role can be changed to Arrived also. In Relocating, robot stops executing its current role and start a new one, and in dynamic environment, robot can become the Leader through competition. Since in its moving, robot's team selection is continually changed and robot may serves different role, thus robot can reassign and relocate its role at any moment.

We prescribe that to the same task, role Relocating abide by the conversion order of Explore->Approach->Attach->Arrived->Leader.

Roles and their dynamically reassigning can be described in figure 2, in which a solid arrow represents Relocating, and dashed arrow represents Dynamical reassigning.

Since Explore does not send but only accept information, after selecting one team Approch only send application of team joining to task Leader, and Arrived does not send out any information, thus by role setting, communication in our system can be reduced largely.

3.3 Attributes related to joint intention making

It is key to decide intension in our five-layer architecture. Below we firstly analyze two kinds of factors that affect the autonomy of task selecting. One kind is indirect influencing factors produced by task, including main task attributes. The other kind is direct factors
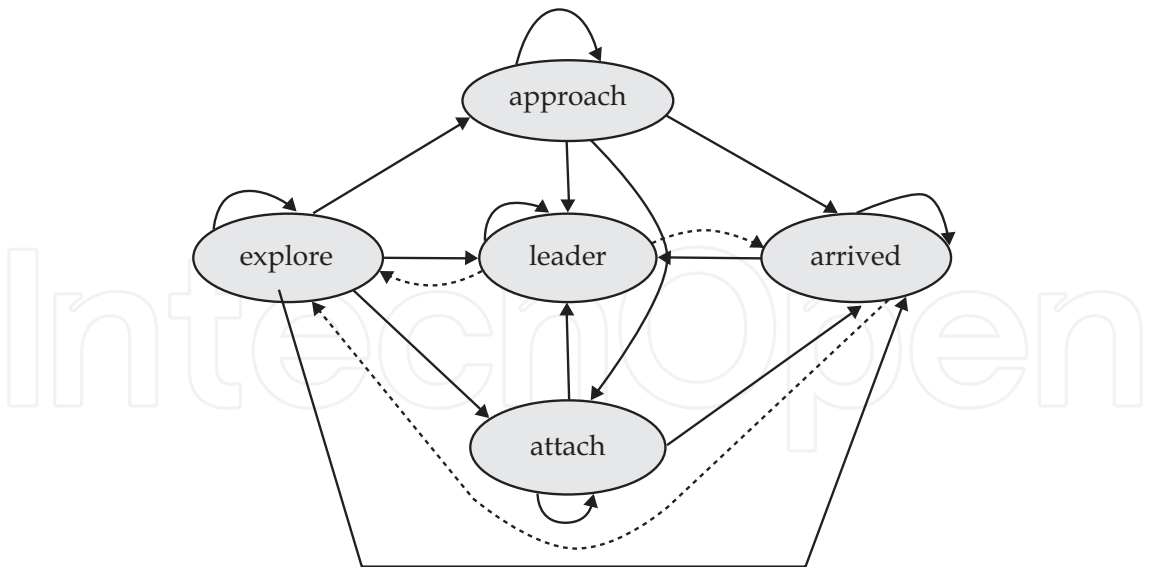
Fig. 2. Depict of role conversion

produced by robot itself, which include robot own capailitis. Below we give details about these two kinds of factors.

1.   Indirect factors

Indirect factors are all kinds of attraction of task to surrounding robot, which including

•   Skills need to fulfill task T_j

Represented by $TCap_j$, is the capability set need for fulfill task T_j. If robot has no $TCap_j$ it will be refused by task T_j.

•   Benefits of task T_j

Represented by UL_j, is the total benefits from fulfilling of task T_j. The more UL_j, the larger attraction of task T_j.

•   Least robot number needed for task T_j

Represented by $Tnum_j$, it the least robot number needed for task T_j.

•   Complex degree of task T_j

Represented by pr_j, complex degree of task T_j is the integrated complex degree, its value is between 0 and 1. In muti-task enviromnet, different tasks have different complex degree.

Value of the above factors are set by system runner, and have no relation to system running time.

•   Location of task

Location of task T_j at time t is represented by $TPlace_j(t)$, which can be fixed or changeable either, for example, when multiple robots are rounding up multiple targets, task location will change continuously.

•   Number of robot taking part in task

Number of robot taking part in task T_j at time t is represented by $Cnum_j$. The smaller value of $Tnum_j$ - $Cnum_j$, the greater attraction of task T_j.

•   Number of robot intend to take part in task

Number of robot intend to take part in task T_j at time t is represented by $Anum_j(t)$. The smaller of $Tnum_j$ - $Anum_j$, the more attraction of task T_j.

•   Task priority

The priority of task T_j at time t is represented by $Pr_j(t)$. The higher of the priority $Pr_j(t)$, the more attraction of T_j.

2.   Direct factors

Direct factors mainly include robot's own effect on task choosing, which include

•   Robot capability

Cap_i is the capability set of robot R_i. If robot wants to take part in a task, it must has the capacity required by that task. Since there are more than one tasks, therefore we can define Q to represent if robot R_i has skills needed by task T_j.

$$Q : Cap \times T \rightarrow \{0,1\}$$

$Q(Cap_i, T_j)=1$ represents robot R_i has skills for task T_j. $Q(Cap_i, T_j)=0$ does not.

•   Success rate

Ab_{ij} is the estimated success rate of robot R_i completing task T_j. As a machinery or equipment, robot will be aging and possibly go wrong, and possibly can not filfill some task. According to its current situation, robot estimate an Ab_{ij}, $0 < Ab_{ij} \leq 1$. The larger Ab_{ij} the more confidence of robot to fulfill task T_j.

•   Task attraction

Task has attraction to robot, which will reduce along with distance increasing[Parker 1999]. Using $Attr_i$ representing the attraction, $distance$ (R_i, R_0) representing distance from Leader robot R_0 to robot R_i, and $0 < distance$ $(R_i, R_0) \leq 1$, then

$$Attr_i = \frac{k}{distance(R_i, R_0)}$$

Where constant k is the largest attraction. In order to discuss conveniently, let k=1. Therefore, $0 < Attr_{ii} \leq 1$. The larger $Attr_i$, the more task attraction.

•   Current state role

According to whether robot R^i is within the communication of R_0 of one certain task T_j, we can set robot R_i to different role, which can be represented by Sta_{ij}. $Sta_{ij}=0$ represents role of Explore, $Sta_{ij}=1$ of Approach, $Sta_{ij}=2$ of Attach, $Sta_{ij}=3$ of Arrived, and $Sta_{ij}=4$ of Leader. When robot select task, it need to fully consider role property, and select the task in which robot's role can be of higher priority.

•   Intending benefit of task completing

$RUL_j(t)$ is benefit expectation of fulfilling task T_j , which can be calculated by

$$RUL_j(t) = \frac{UL_j}{Tnum_j}$$

The larger $RUL_j(t)$, the more interesting robot feeling in task T_j.

•   Communication amount

For data sharing, robots needs communication with each other. Communication amount Com_{ij} is the ratio of the observed robot number (R_v) to the total robot number in the team ($Tnum$), that is

$$Com_{ij} = \frac{R_v}{Tnum}$$

The larger Com_{ij}, the more time cost and less choosing interesting.

•   Interesting factor

Int_{ij} is the interesting degree of robot R_i in task T_j, which can be calculated using

$$Int_{ij} = \frac{Times_{ij}}{\sum\limits_{i} Times_{ij}}$$

Where $Times_{ij}$ is the total times robot R_i had chosen task T_j. From Int_{ij} we can induce the interesting of robot R_i in task T_j, and lead robot R_i to select T_j.

• Time factor

Take into account the continued influence of time interval R_i on the selection of task T_j, we induce a remember coefficient ξ. Denote the interval between that moment and current using Δ time, and according to Ebbinghaus forgetting curve , we can compute the remember rate of robot R_i selecting T_j

$$\xi_{ij} = ce^{\frac{b}{\Delta time}}$$

where b,c are some positive number. The above formula shows that the smaller Δ time, the less possibility of R_i forgetting T_j.

3.4 Decision-making model based on multi-attribute

Decision-making model based on multiple attributes is an important kind of that by multiple rules, which was usually used in fields such as military, economy and polity. This model can be used under condition that parts of tast attributes and weights have been known [Y 2003]. We can descrbe these attributes by a decision-making matrix.

1. Matrix of decision-making based on multiple attributes

Denote the selected attribute set as $Z = \{Z_1, Z_2, ..., Z_m\}$, the task set as $T = \{T_1, T_2, ..., T_m\}$, then the value of m attributes related to n tasks can be represented by a optimal matrix **X**

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1j} & \ldots & X_{1n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ X_{i1} & X_{i2} & \ldots & X_{ij} & \ldots & X_{in} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ X_{m1} & X_{m2} & \ldots & X_{mj} & \ldots & X_{mn} \end{bmatrix}$$

Where X_{ij} represent the ith attribute value of task T_j. all element in matrix **X** are unprocessed primal data, has different physical meaning and usually different dimension, and value difference is larger, so **X** need to be nondimensionalized(or standardization), by transforming actually attribute value in **X** to relative value. To do this, we first divide attributes into benefit type and cost type, according to the influence of attribute value changing on task selecting. Benefit type is that benefit is large if attribute value is, this kind of attribute are also called positive ones. And cost type are that benefit is large and attribute value is small instead, this also be called as invert attributes. Then we can adapt the formula (1) and (2) in [Y 2003] to compute the standardized optimal attribute degree y_{ij} of benefit type by

$$y_{ij} = \frac{X_{ij}}{X_{i\max}}, \qquad X_{ij} \geq 0 \tag{1}$$

and of cost type by

$$y_{ij} = \frac{X_{i\min}}{X_{ij}}, \quad X_{ij} < 0 \tag{2}$$

In (1) and (2), $X_{i\ max}$ and $X_{i\ min}$ are the maximum and minimum of attribute i respectively. After the above standardization, we can get the relative optimal attribute degree matrix Y

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1j} & \dots & y_{1n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ y_{i1} & y_{i2} & \dots & y_{ij} & \dots & y_{in} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ y_{m1} & y_{m2} & \dots & y_{mj} & \dots & y_{mn} \end{bmatrix}$$

2.   Parameter determination

Weight coefficient is very important in our architecture, it can reflect how the attributes act on optimum seeking. By now there are subjective and objective methods to determine the weight. Here we adapt the first, and weight values are set by system runner beforehand, and denote the attribute weight vector w as $w = \{w_1, w_2, ..., w_m\}$, where W_i is the weight of attribute Z_i and w satisifies

$$\sum_{i=1}^{m} w_i = 1$$

3.   Task selecting

In optimal take selecting there is some relativity, the final result is relative to the m attributes of n tasks. Here we take the chose task as a selecting project and for conveniently analysis and comparison, we set a best selecting project Y^+ and a worst one Y^-. In Y^+, all the relative optimal attribute degrees can reach their largest value, and in the worst one, it is on the contrary. Thus we have

$$Y^+ = (y_1^+, y_2^+, ...y_m^+)^T \text{ , and}$$

$$Y^- = (y_1^-, y_2^-, ...y_m^-)^T$$

Where $y_i^+ = \max_{1\leq j\leq n}\{y_{ij}\}, y_i^- = \min_{1\leq j\leq n}\{y_{ij}\}$ are the relative optimal attribute degree of attribute Z_i when select task T_j.

Since the existed n optional projects are commonly between Y^+ and Y^-. So we can not but compare every actual project with Y^+ and Y^-, and to search the nearest to Y^+ and the farmost from Y^-. But it is harder even to do this. Therefore, we introduce a concept of weighted Euclidean distance, and use a relative closing degree to judge the distance from one project to Y^+ and Y^- and select the optimal project. The weighted Euclidean distance of project j to Y^+ and Y^- can be computed

$$D_j^+ = \sqrt{\sum_{i=1}^{m} w_i \left| y_{ij} - y_i^+ \right|)^2}$$

$$D_j^- = \sqrt{\sum_{i=1}^{m} w_i \left| y_{ij} - y_i^- \right|)^2}$$

Where w_i is the attribute weight. Thus the relative closing degree of project j is

$$C_j = \frac{D_j^-}{D_j^+ + D_j^-} \qquad\qquad j = 1, 2, ..., n$$

It is clearly that $0 \le C_j \le 1$, and the larger C_j, project j be nearer to Y^+ and be better. As to Y^+, $C_j = 1$, and to Y^-, $C_j = 0$.

3.5 Behavior decision-making algorithms based on multiple attributes

Behavior Decision-Making in dynamic environment has properties such as robot's own autonomy and isomerism, variability of the dynamic environment, distribution of task, and task's own isomerism and variability. Thus in task executing, robots need to select a fitter task according to the exterior and interior on-line information, and make the whole system be in a changefully task assigning state. And just because of this, robot will dicover more than one task at one moment, and it must chose one team to participate in according to its knowledge about the task and by the judging of its own capability.

Recently, most algorithms for task choosing merely have considered whether robots fits task, or only have further considered the single attribute of distance from robot to task. In this research, according to analysis in the last section and at the precondition robot suit to task, by fully considering many factors such as task, environment and robot itself, combing attribute weight, and we propose a new robot behavior decision-making algorithm.

The main idea of our algorithm is that robot use greedy strategy to choose task, this also was called the one-step optimization. The algorithm idea is that, after every time interval Δt, in order to obtain information about other robots' current situation, tasks needed to perform, and situation of known tasks, robot will exchange with other robots within its communication range. Integrating the information, robot will firstly find out a set of tasks it competent for, then determine the relative attributes and weights, load the multiple attributes decision matrix, and at last choose target task. These details are shown in figure 3.

In the implement, robot state in system can be described by a septuple

$$R =_{def} < id\_robot, Cap, State, Envir, Rec, Comm, Time >$$

Where

Id_robot is the unique ID of robot;

State is the current interior state of robot, which is represented by a quintuple (Sta, Vic, θ, id_ros, $SRole$), here Sta is robot's location, Vic and θ are robot's speed and direction of current moving, id_ros is the identifier of robot's chose task, and $SRole$ is robot's current role. id_ros=0 represents that robot has not selected any task , accordingly $SRole$ ='Explore';

Cap represents robot's capability, which can be described by a group of ($SRange$, $CRange$, MV, AC, ST, $REAC$, CO, PL, OR, LE), which are robot's vision radius, communication radius, and the ability of moving, behavior, behavior decision-making, real time reaction, cooperation, local programming, organization, and of learning respectively;

Envir is the state space set of robot's possible world, such as information of surrounding geographical environment, of obstacles and surrounding robots;

Rec are history records, which are combined by a tetred (T_j, $TSta_j$, Pre_j, $time_j$) of task T_j, where $TSta_j$ is the latest state, Pre_j represent the predecessor robot related to task T_j, that is the *ID* of the robot who apprize the latest information, and $time_j$ represents time when the robot come into learn task T_j;

Exchange with all other robots within the communication range to obtain a new task set T1

Merge new task set T1 with the older task set.
T=T+ T1

Initial: i=1; M=|T|

Judge each T_i in T

i=i+1

If robot R_j is competent for T_i

yes

No

T=T-{t_i}; i=i+1

No

i>M

yes

m=|T|;
Robot R_j determine the affect factor according to actual situation
$Z_1, Z_2 ... Z_n$

Construct optimal selecting matrix X, and compute standardized optimal attribute degree matrix Y

Determine attribute weight, and the best and worst projects

Compute relative closing degree C_j of each T_j

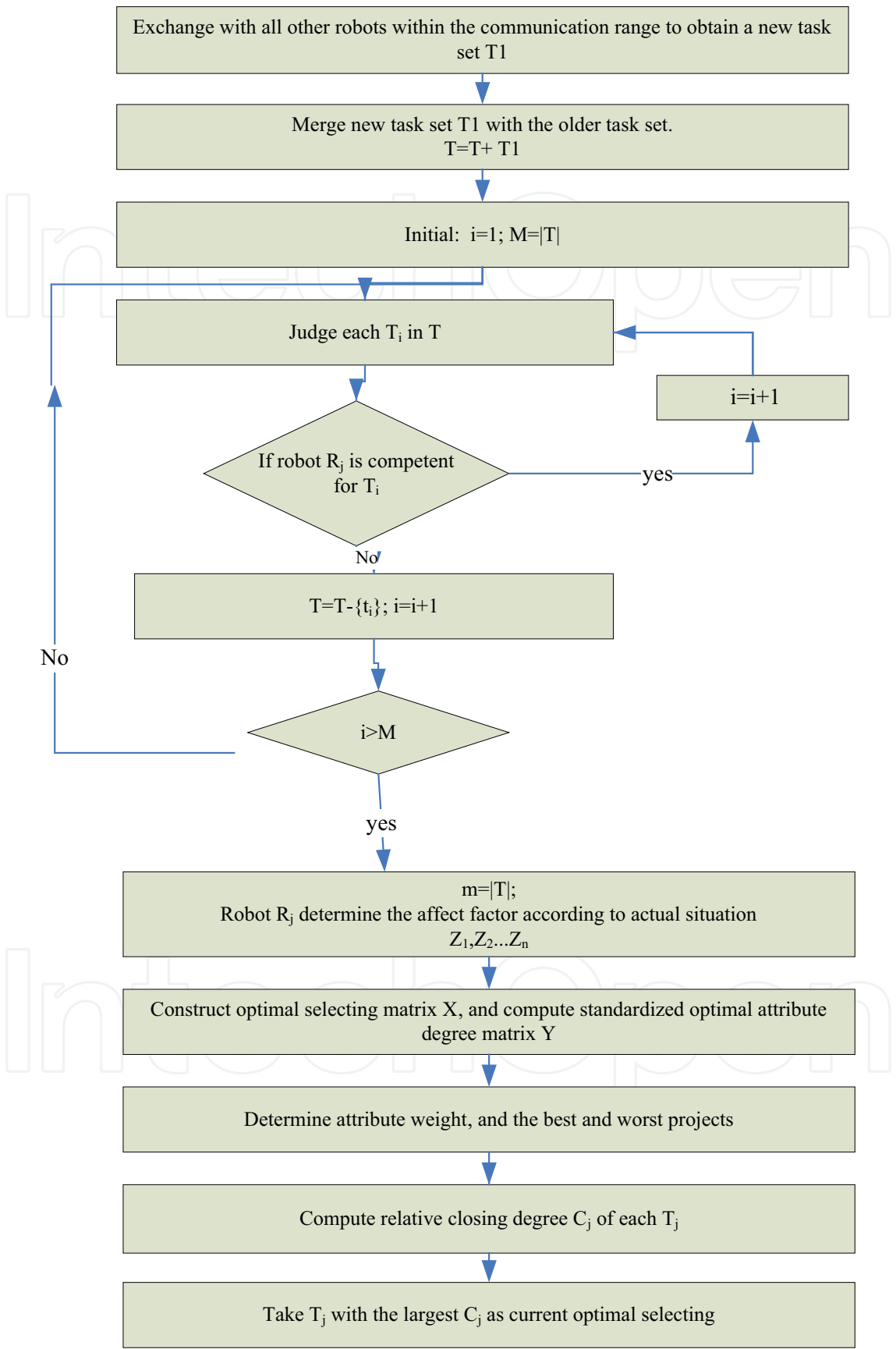Take T_j with the largest C_j as current optimal selecting

Fig. 3. Flow chart of behavior decision-making algorithm based on multiple attribute

Comm is robot communication language; And

Time is the time of updating record, which used in maintaining robot group.

The Algorithm in detail is described in below algorithm 1.

Algorithm 1: Behavior decision-making algorithm based on multiple attribute

/*Algorithm description: After every time interval Δt, robot R_i will update organizing or coordinating database and task information perceived by itself or obtained by communication, and choose one task*/

Step 1.   Set i, $1 \le i \le m$

Step 2. R_i exchange with robot R_i within its communication range.

Step 3.   if R_i find a new task in its vision range, then $R_i.Rec.Rv_i \leftarrow$ Leader;   /*set R_i the Leader of the new task*/

Step 4. R_i judge if there are any other new tasks by analysis the received information, if there not any, R_i maintain its previous situation and goto 15. /*to maintain task's relative stability*/

/* below steps dealing with currently known tasks*/

Step 5.   merge new task with the previous task set, set the counter count $\leftarrow 1$, and set the known task number to *M*.

Step 6.   according to robot's selected task, divide all robots into M class

Step 7.   to each task, judge if it has been completed. if has, delete its information and goto 10.

Step 8.   if the new task k is the same as the current task of R_i, then update the information of task k with its latest information. Set the *ID* of robot who giving new task information with u, and goto10

Step 9.   if $R_i.Cap \subseteq T_j.TCap_j$, then add task T_j, add set *ID* of robot who giving new task information with u

Step 10. count←count+1

Step 11. if $count\delta \le M$ then goto 7

Step 12. call the behavior decision-making algorithm based on multi-attributes, find the task whose intension function get the largest value by calculating $\max(Ui_j(t))$.

Step 13. if $k == R_i .id\_ros$, then goto 15 /* to perform the same task as current*/

else R_i send a request to robot R_u for joining in task k.

Step 14. if R_i be accepted by task k, then set $R_i.id\_ros = k$.

Step 15. if $R_i .id\_ros$<>0 , then calculate the distance from robot R_i to Leader, and reassign R_i's role, then call the behavior programming module. Start and perform the behavior control module.

Hereon, robot uses a one-step optimal searching strategy to choose task. This has merits such as below

- After each moving step, robot will collect new information about tasks and environment, to avoid information outdated.
- Can find disabled robot. If one robot has not updated its information for a long time, then other robot take it being wrong and will inform this to Leader.
- By using the minimized task relationship, potential communication conflicts and time needed for conflicts solving can be reduced, thus system efficiency can be improved. In Algorithm 1 each step is the optimal, so it is fit for changeful dynamic environment in which it is difficult to make a wholly route programming.

4. Garbage disposal in dangerous environment and its implement

In this section, in a simulated cross-country environment, test result of multiple robots fulfilling garbage disposal through efficiently corporation was used to validate the efficiency of the proposed hierarchical controlling architecture and the decision-making algorithm.

4.1 Problem description

Multiple robots cooperatively performing garbage disposal was a loosely coupling problem, that is, robots separately search and dispose the garbage. There are two kinds of method to do this. In the first kind of method, robots of different labor division will cooperatively dig and bury garbage on the spot. In the other kind, robots firstly need to convey garbage to some specified center then to dispose. Here we take the two kinds as two works.

In system, there set three class of robots, in which class i are crane robots adept at pushing and convey object. Class ii are rooter robots adept at sapping. And class iii are conveyance robot. All the three kinds of robot have stored information of task, can perform functions such as task searching, organize, and coordinate. the first kind of task needs corporation of robot class I and ii, in which after robot of class ii fulfilling dig, one or more than one robot of class i will push garbage into the hole. And the second kind of task need the corporation of ii and iii, as that robot of class ii grasp and transfer garbage to robot class iii, and one or more robot of class iii will transfer garbage to an appointed location .

At system beginning, robots having capabilities of environment perceiving and behavior programming were dispersed randomly in the cross-country environment, and began to search garbage object. When find some garbage, robot will firstly judge whether it can dispose the garbage by itself, if can, then began to, otherwise it will take the garbage as a task and itself as the Lead, make sure numbers of three class robot needed according to task's size and property, and began to inform other robot to form a group. Without loss of generality, we assumed that task of the first kind need 1 robot of class ii and a few of class i, when robot number meet the required, robot of class ii will surround the task, which represent robots have disposed garbage jointly, and task has been completed. As to the second kind of task, we assume there need 1 robot of class i and a few of class iii, similarly all the needed robots must firstly come to the garbage, then they form a column team for garbage portage. After convey garbage to a target location, the task be completed.

In realization, the number of needed robot and of that around were labeled in a square bracket beside the task. Three colors as we listed above were used to indicate the founded, being performed and completed state of task respectively. If task is hung, then its needed robot number is 0.

4.2 Implement strategy

Form point of view of organization, robots can be divided into task Leader and collaborators. As manager of task, Leader will fulfill certain task also. Leader uses algorithm 2 below to build up a team, and also to fulfill task as soon as possible.

Algorithm 2: Leader select team member

Step 1. After tasks being announced, Leader begin to time and set i=0;

Step 2. Leader announces current tasks information to robots within its communication;

Step 3. Leader judge each answered robot R_i if it has competent to join the team. If it has not, goto 8;

Step 4.   Leader read the capacity of R_i related to task T_j;

Step 5.   if $Tnum_j(t) \geq Cnum_j(t)$, then announce the end of recruit, and goto 7

Step 6.   announce robot R_i as one of pre-team member, and if $R_i.SRole = Arrived$ then $Cnum_j(t) = Cnum_j(t) + 1$; else if $R_i.SRole = Attach$ then $Anum_j(t) = Anum_j(t) + 1$;

Step 7.   if there be $Tnum_j(t) \geq Cnum_j(t)$ (j=1，2，…，n), then goto 10;

Step 8.   if it has not exceed the scheduled time, then i=i+1, and gotto 3;

Step 9.   Leader announces the fail of formation;

Step 10. Leader determines final team members need for the task, and notifies all the selected members.

In system running, each robot corresponds to one of the five state roles. For a relative stable team, and reducing the unneeded cost of role exchanging, here we prescribe that only Explore, Approach and Attach can bid and select team, robot after having chosen task only send application to Leader of the task , and only Leader,Approach,Attach can announce their knowledge of tasks.

In system we abstract and define task as class, in which we emphasize task needed robot number, and the change of the number. The task class can be described as

```
class TARGET
{ CPoint m_Position;        //task location
  int m_nValue ;            //benift obtain from task completing
  int m_nComplax ;          //complex degree of task fulfilling
  int m_nPrior  ;           // priority of current task
  int m_type;               //kind of current task, m_ type =1 represent bury on the spot,
                            // m_type =2 represent transfer to a centralized spot then bury.
  int m_nNeedRobotNum1;        // needed robot number of type  I for task fulfilling
  int m_nNeedRobotNum2;        // needed robot number of type II
  int m_nNeedRobotNum3;        // needed robot number of type III
  int m_nArrivedRobotNum1;     // arriveded robot number of I
  int m_nArrivedRobotNum2;     // arriveded robot number of II
  int m_nArrivedRobotNum3;     // arriveded robot number of III
  int m_nConfirmRobotNum1;     // robot number of type I chosen the task
  int m_nConfirmRobotNum2;     //robot number of type II chosen the task
  int m_nConfirmRobotNum3;     //robot number of type III chosen the task
  int m_nLeaderId;          // Leader ID
  bool m_bfinished;         //if task fulfilled
  Real BeginTime;           //task started(or discoveredd) time
  Real EndTime;             // task completed time
  CList<Cpoint, CPoint&> m_RoundPoint;   //coordinate of points surrounding the task.
      };
```

And robot in the system is another class, which is

```
class Robot
 { int RobotId;             //robot  ID
   Type m_nType;            //number represents robot's current state role, 0 as Explore, 1
                            //as Approach, 2 as Attach, and 3 as Leader
   Type Cap;                 //robot's capacity type
   int m_nViewDistance;     //robot's vision range
   int m_nComDistance;      //robot's communication range
```

```
CPoint m_Position;        // robot's current location
int m_nSpeed;             // robot's speed
int m_nTargetId;   //robot's choosing task, value -1 represent it has not chosen any task.
int m_nMaxValue;                    // upper limit of task benefit estimated by the robot
int m_ObjectTarget[1501][1501];  //robot's known information about task, value -2
                          //represent no task , >=0 represent discovered some task,
                          //value -1 represent not discovered any task, and -3 represent
                          // having completed the task
Char  m_Pa[n];            // information about robot's known teammate
Real CuTime;              //time of current record
};
```

4.3 Implement strategy

In our system, robot adapt the controlling based on behavior, and behaviors can be divided into four kinds as below

• move_to_goal;

• maintain_formation, after chosen one task and but can not see the task directly, robot take this behavior lest diverge from the main direction;

• avoid_static_obstacle, robot takes this behavior lest it collides with obstacle;

• avoid_robots, robot takes this action lest it collides with other ones.

After chosen one task team, robot take that task's location as its moving target, and Leader will assign one role to it, this role correspond to robot location after task fulfilling. And after all robot members determined their task's location, the moving and coordination in robot team can be taken as the processing of team formation. Each robot need to determine its team's formation vector according to its surrounding condition, such as the location of task and other members, and the surrounding obstacle. Referring to formation vector constructing algorithm in [Dong et al. 2000; Chio et al. 2003; Balch & Arkin 1995; Balch & Hybinette 2000; Han 2003], we first prescribe some identifiers such as $R = \{R_1, R_2, \cdots R_n\}$ is a robot set with n members, $TARGET = \{T_1, T_2, \cdots T_n\}$ is a task set, $OBSTACLE = \{O_1, O_2, \cdots O_m\}$ is an obstacle set, and d_i is formation vector by which R_i can control its cooperation with other members thus to maintain the formation. Value in d_i can indicate robot R_i's driving power and moving direction.

By the above identifiers, Algorithm 3 of formation controlling during moving and coordination can be described as below.

Algorithm 3: Formation controlling algorithm

Step 1. Determining robot type. In the system robots are dividend into type **A** that can see the task directly, type **B** can see task indirectly, that is robot although can not see task itself, but can see other robots in its team, and type **C**, can not see task or any other robots.

Step 2. Using formula (3) to determine formation vector d_i

$$d_i = \begin{cases} K_1 Q_i & R_i \in A \bigcup B \\ K_2 Q_i^{'} & R_i \in C \end{cases} \tag{3}$$

Where K_1, K_2 are two controlling parameters, we will give their value in the simulation test. Q_i is a unit vector from robot R_i to other tasks or members, and $Q_i^{'}$ is a unit vector from R_i and normal to Q_i.

Step 3. judge if there are some obstacles or team members along each robot's formation vector d_i , and using result of formula (4) to weightedly sum up every robot's sub behavior, thus to modify their moving directions.

$$\begin{pmatrix} x_i^i \\ y_i^i \end{pmatrix} = \alpha\tau_t K_1 \begin{pmatrix} x_t^i \\ y_t^i \end{pmatrix} \Big/ \left\| \begin{pmatrix} x_t^i \\ y_t^i \end{pmatrix} \right\| + \beta\tau_{j,j\in R} K_1 \begin{pmatrix} x_j^i \\ y_j^i \end{pmatrix} \Big/ \left\| \begin{pmatrix} x_j^i \\ y_j^i \end{pmatrix} \right\|$$
$$+ \gamma K_2 \begin{pmatrix} x_o^i \\ y_o^i \end{pmatrix} \Big/ \left\| \begin{pmatrix} x_o^i \\ y_o^i \end{pmatrix} \right\| + \sum_{k\in R\cup OBSTACLE} \delta_{ik} \left\{ \begin{pmatrix} x_k^i \\ y_k^i \end{pmatrix} - L \begin{pmatrix} x_k^i \\ y_k^i \end{pmatrix} \Big/ \left\| \begin{pmatrix} x_k^i \\ y_k^i \end{pmatrix} \right\| \right\}$$
(4)

In (4), $\left(x_i^i, y_i^i \right)^T$ be R_i 's moving direction during the current controlling cycle T_c . Coefficient α, β and γ can be

$$\begin{cases} \alpha = 1 & \beta = 0 & \gamma = 0, & R_i \in A \\ \alpha = 0 & \beta = 1 & \gamma = 0, & R_i \in B \\ \alpha = 0 & \beta = 0 & \gamma = 1, & R_i \in C \end{cases}$$

τ_i be the attraction force of task T_j on robot R_i , τ_j be inducing force of other robot R_j on R_i , and δ_{ik} be the exclusive force of obstacles or other robot on robot R_i . If distance from R_i to other robot or obstacle is less than some L, then $\delta_{ik} = \delta > 0$, otherwise there is no exclusive force, that is $\delta_{ik} = 0$.

Step 4. estimate robot R_i 's $Position_i$ at next moment.

Step 5. R_i moves to $Position_i$. If $|Position_i\text{-}T_i| < \varepsilon$, then R_i directly arrived at task location T_i , and goto 2.

4.4 Simulation plateform

1. Simulated environment

The system uses a 1501×1501 pixels 2-dimensional cross-country simulation environment, which represent a 500×500 meters actual environment, value of pixel represent land-use type of actual environment, which can be road, lake, river, natural land Surface, fortification, sandlot, and plate. Surface such as grass and road can get through, and diked area such as lake, road, and fortification can not.

2. Development tool

We use Microsoft VC++ 6.0 as the development IDE.

3. Running plate

Our simulation need to run on operating system of Windows 2000/XP.

4. Functions of main module

The main function of out simulation software includes

- Display environment map with depth feeling of environmental information
- Map can zoomes in or out
- Provides convenient interface for system user
- Dynamically and intuitively shows the process of robot formation for task fulfill.

5. Contents in display area in program running

In our system running, robot type of Crane, Rooter and Transporter distribute randomly, and numbers of the three type robots are approximately equal. The three type robots are represented by symbols as

■ Crane
■ Rooter
■ Transporter

Two kinds of tasks are represented by circle and ellipse respectively, that is the 1'th task type using circle, and the second using ellipse. And according to time order, tasks are divided into three types, which we using three colors to express, that are

● Undiscovered task

● Discovered but not completed task

■ Completed task

6. Status area

In status area below display area, there displays the location, needed robot number, having recruited number of current task, and system running time.

4.5 Simulation of formation based on task

At the beginning, a few of tasks and robot randomly distribute in the cross-country environment. We suppose that robot know their location and can communicate faultlessly. Every robot has its own range of communication and vision and speed. And commonly communication range is large than that of vision, and vision range is larger than speed×per step time. System controlling parameters are set as in table 1. In the below demonstrating map, we use color to represent robot role conversion, and beside each task there is a square, in which robot number needed for task, and already recruited number are listed. Task such as being discovered, executed and completed are distinguished by different colors, and states being hang uses 0 needed robot number to express. the six below demonstration map show the whole formation process, including produce, building, programming and executing of cooperation.

Figure 4 is initial state of system running. There randomly distributes 6 tasks, represented by character of A to F, and 17 robots including 6 crane robots (in color of pink), 6 transfer robots (in red) and 5 rooter robots (in black).

Figure 5 is the system after running 1.75 s. Robot found 4 tasks including C, D, E and F, and task F needed transferred. In square bracket beside each task there labeled the number of robot needed and that of arrived.

Figure 6 shows system after running 2.406s. Task C has already been fulfilled, and task A and B been founded. Task B was hang, so its needed robot number is 0, two robots around B are moving towards other tasks. Task D needed one robot but around it there are two, the redundancy is mainly caused by errors in communication and time setting.

Figure 7 shows system after running 3.156s. Tasks A~D have been completed, robot number of task E already is 3, but that of task F have not changed. The black robot moving to task F found that F out of its capability, so left it to other task.

Figure 8 shows system after running 10.327s. Robot number of task E is 4, and that of task F unchanged, no other robot move to task F, thus deadlock came into being.

Figure 9 E and F shorting of robot, and within their communication no available ones, deadlock occur.

4.6 Result analysis

1. Four strategies in the system

In the system we use four controlling strategies

T0, integrated controlling strategy combining autonomic and entire controlling model

Fig. 4. Initial system state

Fig. 5. System state after running 1.75s

Fig. 6. System state after 2.406s

Fig. 7. System state after 3.156s

Fig. 8. System state after 10.327s

Fig. 9. E and F occur dead-lock

T1, integrated controlling strategy combining random and entire controlling model
T2, integrated controlling strategy combining autonomic and partial controlling model
T3, integrated controlling strategy combining random and partial controlling model.

Here autonomic controlling is that robot adapting behavior decision-making algorithm to select behaviour (to decide its role and team) based on multiple attributes proposed in this chapter, making using of information by watching by exchanging. Randomly controlling is that robot randomly select one team to join. And entire controlling is that in system there is a virtual robot, it can collect information of all the known tasks, including task location, task needed robot number, task's recruited robot number and arrived robot number, and set priority level to each task thus to induct robot's team selecting. Virtual robot also will check at intervals if dead-lock occurring (when all task's robot number have not satisfied but no robot in state of Explore, or all the remained robots are at state of Explore), if occurring, virtual robot must solve it by forcing some robot to take one role or take part in certain team. Interval of entire controlling will be determined by dead-lock occurring frequency at last time, if this frequency is high, then shorten the interval, otherwise, extend it.

Success times in our simulation are the times robot complete task by its own coordination without virtual robot's intervening. And completing number is robot's completing tasks under entire controlling plus limited intervening.

2. System parameter setting

To valid the proposed architecture in this chapter, here we set value of all the common parameters in table 1, other special parameters value of simulation will be set in respective test table. In table 1, unit of running time is ms (millisecond), and units of communication and vision range are pixel.

Parameter identifier	Parameter value	Parameter meaning
Range	40	Surrounding radius
τ_i	1	Attraction of task T_j on robot R_i
τ_j	1	Inducing power of robot R_j on R_i
K_1	20	Coefficient of attraction and inducing power
K_2	100	Coefficient of exclusive power
δ	100	Exclusive coefficient of obstacle to robot R_i

Table 1. System controlling parameter

3. Result and analysis

We give tests for five kinds of comparisons as below.

Test 1, test with same task number and different robot number, result is listed in table 2.
Test 2, test with same robot number and different task number, results is listed in table 3.
Test 3, test with different vision range, result is listed in table 4.
Test 4, test with different communication range, result is listed in table 5.
Test 5, Test with different time length per running step, result is listed in table 6.

Reasult in Table 2 shows that along with the increasing of robot numbers, total time needed for task fulfilling decreased, numbers of discovered tasks and completed tasks increased, and dead-lock number lessened.

Basic parameter value					
Robot number	Task number	Most robot number needed for task	Total loop number	Total time / time per step	Vision range / communication range
10	5	6	50	20000/500	50/100
Test result					
	Total time consume	Success times	Discovered task number	Completed number	Dead-lock number
T0	1075.454	12	163	168	435
T1	1110.5.2	8	184	122	496
T2	1058.156	16	180	156	304
T3	1111.434	8	178	118	476
Basic parameter value					
Robot number	Task number	Most robot number needed for task	Total loop number	Total time / time per step	Vision range / communication range
20	5	6	50	20000/500	50/100
Test results					
	Total time consume	Success times	Discovered task number	Completed number	Dead-lock number
T0	1193.84	14	217	185	289
T1	1071.517	15	225	224	174
T2	1473.213	19	240	213	223
T3	1565.595	2	224	133	220
Basic parameter value					
Robot number	Task number	Most robot number needed for task	Total loop number	Total time / time per step	Vision range / communication range
30	5	6	50	20000/500	50/100
Test result					
	Total time consume	Success times	Discovered task number	Completed number	Dead-lock number
T0	554.454	23	236	216	56
T1	652.22	23	240	230	72
T2	341.4	28	244	226	69
T3	305.233	28	241	227	83

Table 2. Results of test with same task number and different robot number

Value of basic parameters					
Robot number	Task number	Maximum needed robot number	Total loops	Total time / time per step	Vision range/ communication range
20	3	6	50	20000/500	50/100
Test result					
	Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
T0	1179.749	26	137	126	88
T1	1140.765	23	136	121	194
T2	1084.253	36	138	136	78
T3	1066.696	37	137	137	96
Value of basic parameters					
Robot number	Task number	Maximum needed robot number	Total loops	Total time / time per step	Vision range/ communication range
20	4	6	50	20000/500	50/100
Test result					
	Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
T0	821.091	29	170	164	197
T1	1253.700	25	163	163	348
T2	1122.657	29	165	151	246
T3	1122.08	33	167	152	198
Value of basic parameters					
Robot number	Task number	Maximum needed robot number	Total loops	Total time / time per step	Vision range/ communication range
20	5	6	50	20000/500	50/100
Test result					
	Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
T0	1193.84	14	217	185	289
T1	1071.517	15	225	224	174
T2	1473.213	19	240	213	223
T3	1565.595	2	224	133	220

Table 3. Test results with same robot number and different task number

When recruited robot number far exceeds the needed number, having or hving not used entire controlling, the numbers of dead-locks and completed tasks is not different. The reason is that when robot number is large, possibility of dead-lock will reduce, then even not using the entire controlling, robot can fulfill task quickly by autonomic controlling.

If no more robot available, then although using entire controlling and bid algorithm based on intension, because of there are many weak intervening, which will reduce robot's autonomic capability, so robot only can discover less tasks. And if not using entire controlling intension-based algorithm, then system had to solve more dead-locks, thus will reduce robot's task discovering capability too.

Reasult in Table 3 shows that the more the task, the more the dead-lock, and the less completed task number and the higher time consuming.

Value of basic parameter		Test result					
Robot number	20						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	1179.749	26	137	126	88
Total loops	50	T1	1140.765	23	136	121	194
Total time/ time per step	20000/ 500	T2	1084.253	36	138	136	78
Vision range/ communication range	50/100	T3	1066.696	37	137	137	96
		Test result					
Robot number	20						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	506.72	14	64	64	74
Total loops	50	T1	553.316	12	62	62	91
Total time/ time per step	20000/ 500	T2	937.531	6	52	52	84
Vision range/ communication range	25/100	T3	702.733	5	40	40	74

Table 4. **Test** result of different vision range

Value of basic parameter		Test result					
Robot number	10						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	774.366	28	123	118	317
Total loops	50	T1	1054.996	16	106	82	536
Total time/ time per step	20000/ 500	T2	849.47	22	114	108	398
Vision range/ communication range	50/100	T3	1125.786	18	100	78	492
		Test result					
Robot number	10						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	839.421	16	114	107	422
Total loops	50	T1	950.795	7	107	75	182
Total time/ time per step	20000/ 500	T2	757.423	20	106	100	206
Vision range/ communication range	50/200	T3	750.513	21	110	88	425

Table 5. **Test** result of different communication range

Reasult in Table 4 shows that vision range is better to be large.

Reasult in Table 5 shows that as to communication range it is not the greater the better. The decreasing of completed task number along with the increasing of communication range in the result has illustrated this point.

Reasult in Table 6 illustrates that changing of time per step has no more influences on task fulfilling. But if adopting the entire controlling and intension-based bid recruiting algorithm jointly under a moderate ratio of robot number to total task number, then the total time consuming will reduce remarkably, and success time will increase remarkably.

Value of basic parameter		Test result					
Robot number	10						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	774.366	28	123	118	317
Total loops	50	T1	1054.99	16	106	82	536
Total time/time per step	20000/500	T2	849.47	22	114	108	398
Vision range/communication range	50/100	T3	1125.78	18	100	78	492

		Test result					
Robot number	10						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	787.652	24	122	112	300
Total loops	50	T1	1121.10	14	110	80	462
Total time/ time per step	20000/700	T2	893.756	22	114	104	404
Vision range/ communication range	50/100	T3	1086.30	17	94	75	502

		Test result					
Robot number	10						
Task number	3		Total time consuming	Success times	Discovered task number	Completed task number	Dead-lock number
Task's Maximum needed robot number	6	T0	798.72	28	124	120	296
Total loops	50	T1	1091.99	14	106	80	398
Total time/ time per step	20000/1000	T2	884.734	24	114	106	250
Vision range/ communication range	50/100	T3	1102.62	18	100	78	368

Table 6. **Test** result of different running time per step

4.7 Simulation conclusion

From the above comparisons we can found that the discovered and completed task number, and total time consuming are mainly affected by factors such as the adapted strategy, ratio of robot number to task number, range of vision and communication, and system total running time. Generally speaking, we can have these four conclusions

1. When existed robot number does not exceed the total needed robot number of all tasks, result by jointly adapting entire controlling and intension-based bidding algorithm is better than that neither strategy being used. But if the existed robot number is larger, adapting those two strategies will make system more complicate and less efficiency.
2. Using entire controlling and intension-based bidding algorithm jointly will induce some weak intervening. But if not using neither of the two strategies, then time consumed in dead-lock solving will increase, which will also reduce robot's efficiency.
3. Along with the increasing of robot number, task numbers discovered and completed will increase, and dead-lock number and total time consuming will decrease remarkably.
4. In searching, along with the increasing of vision range, number of completed tasks will increase largely. But along with increasing of communication, it is on the contrary.

So for the sake of efficiently fulfilling given tasks, the practicable solution is that besides jointly adapting entire controlling and intension-based bidding algorithm, we can set robot number the half of total needed robot number, and extend robot's vision as large as possible.

5. Conclusion

In this chapter, we firstly gave analysis on some typical architectures of robot system, and took for that in multi-robot system that oriented to task under a dynamic environment, the prominence should be given to robot capabilities of self-adopting, real time reaction, behaviour autonomic decision-making, and cooperation, especially to behavior autonomic selecting, but all this has not presented in the existed architectures. Therefore, we proposed a hybrid hierarchical controlling architecture of five layers, in which behavior decision-making as an independent module is expressed. And then we emphatically studied the implement of the behavior decision-making module. And at last, we used the effectively fulfilling of garbage disposal by robot's corporation in cross-country environment to validate the proposed hierarchical controlling architecture and decision-making algorithm. Simulation results showed that our architecture and algorithm are effective.

6. Acknowledgment

7. References

Balch, T. & Arkin, R C.(1995). Motor Schema-based Formation Control for Multi-agent Robot Teams, *in Proceeding of the First International Conference on Multi-agent*

Systems, pp.10-15, ISBN 978-0262621021, San Francisco, June,1995, The MIT Press, Cambridge, Massachusetts (USA).

Balch, T & Hybinette, M. (2000). Social Potentials for Scalable Multi-Robot Formation, *IEEE International Conf. on Robotics and Automation*, pp. 73-80, ISBN 978-0780358867, April, 2000, IEEE, New York, NY (USA).

Caloud, P.; Choi, W., Latombe, J C.; Pape, C L. & Yim, M. (1990). Indoor automation with many mobile robots, *Proceeding of IEEE international Workshop on Intelligent Robotic system*, pp. 67-72, ISBN 90-247-3346-4, Ibaraki, Japan, July, 1990, IEEE, New York, NY (USA).

Cao, Z Q.; Zhang, B. & TAN, M. (2001). Individual Control Architecture for Multiple Robot System. *Robot*, Vol. 23, No. 5, 450-454, ISSN 1002-0446.

Chaimowicz, L.; Campos, M F M. & Kumar, V. (2002). Dynamic Role Assignment for Cooperative Robots, *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 293-298, ISBN 0-7803-7272-7, Washington,May, 2001, ,IEEE, New York, NY (USA).

Yue, C Y.(2003).*Decision-making Theory and Methods*, Science Press, ISBN 7-03-010816-7, Beijing, China.

Chio,T S. & Tarn, T J. (2003). Rules and Contral Strategies of Multi-Robot Team Moving in Hierarchical Formation. *IEEE International Conf. on Robotics and Automation*, pp. 2701-2706, ISBN 0-7803-7737-0, Taipei, Taiwan, September, 2003.

Dias, B. (2004). Trader Bots: *A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*, the Robotics Institute, Carnegie Mellon University, 2004.

Dong, S L.; Chen, W D. & Xi, Y G.(2000). Distributed Control System for Multi-robot Formation. *ROBOT*, Vol. 20, No.6, 433-438, ISSN 1002-0446.

Farinelli, A.; Scerri, P. & Tambe, M.(2003). Building Large-scale Robot Systems: Distributed Role Assignment in Dynamic, Uncertain Domains, *In AAMAS'03 Workshop on Resources, role and task allocation in multiagent systems*, ISBN 1-58113-683-8, Melbourne, Australia, July, 2003, ACM, New York, NY (USA) .

Habib,M K.; Asama,H. & Ishida Y.(1992). Simulation Environment for an Autonomous and Decentralized Multi-agent Robotic System, Proc. IROS'92, pp. l550-1557, ISBN 0780307372, Raleigh , NC (USA) , July 1992, IEEE, New York, NY (USA).

Han, X d.; HONG, B R. & MENG, W.(2003). Distrubuted Control for Generating Arbitrary Formation of Multiple Robots. Robot, Vol. 25, No. l, 66- 72, ISSN 1002-0446.

Laengle, T.; Lueth,T C.; Rembold,U. & Woern, H. (1998). A Distributed Control Architecture for Autonomous Mobile Robots-implementation of the Karlsruhe Multi-agent Robot Architecture (KAMARA). *Advanced Robotics*, Vol. 12, No.4, 411-431, ISSN 0169-1864.

Noreils,F R.(1993). Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environmentm. *The International Journal of Robotics Research*, Vol. 12, No. 1, 79-98, ISSN 0278-3649.

Parker, L E.(1998) *ALLIANCE:* An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, Vol. 14, No.2, 220-240, ISSN 1042-296X.

Parker, L E.(1999). Cooperative Robotics for Multi-Target Observation. *Intelligent Automation and Soft Computing*, *special issue on Robotics Research at Oak Ridge National Laboratory*, Vol. 5 , No.1, 5-19, ISSN, 1079-8587.

Tambe M. & Zhang W.(1997). Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*(JAIR), Vol. 7, No.1, 83-124, ISSN 1076 - 9757.

Tan, M.; Wang, S. & Cao, Z Q.(2005) *Multi-r*obot systems, Tsinghua University Press, ISBN 7302100950 , Beijing, China.

Tang, Z M. (2002) *Research on Essential Techniques for Mobile Intelligent Robot and Robot Team* [D].Nanjing university of science and technology, Nanjing, Jiangsu province, China.

Xu D. (2004). Research on Some Key Techniques for Multi-robot System. Applied Science and Technology. Vol. 7, No. 31, 37-39, ISSN 1009-671X.

Zhao, Y W. & Tan, D L. (1990). Study of Robot Architecture in Multiple Mobile Robots System. *Robots*, Vol. 21, No.6, 421-425, ISSN 1002-0446.
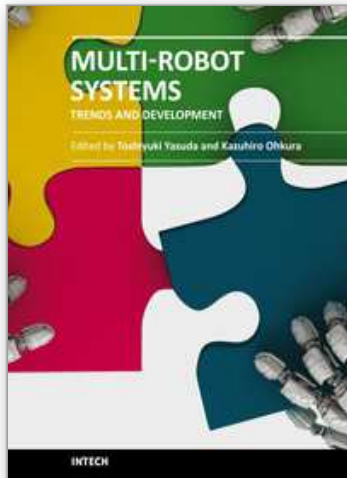
Multi-Robot Systems, Trends and Development

Edited by Dr Toshiyuki Yasuda

This book is a collection of 29 excellent works and comprised of three sections: task oriented approach, bio inspired approach, and modeling/design. In the first section, applications on formation, localization/mapping, and planning are introduced. The second section is on behavior-based approach by means of artificial intelligence techniques. The last section includes research articles on development of architectures and control systems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

INTECH
open science | open minds