# Partner selection mechanisms for agent cooperation

Toktam Ebadi, Maryam Purvis, Martin Purvis
*University of Otago, Dunedin, New Zealand*
*E-mail:{tebadi, tehrany,mpurvis}@infoscience.otago.ac.nz*

## Abstract

*In a multi-agent system, a single agent may not be capable of completing complex tasks. Therefore agents are required to form a team to fulfill the task requirements. In this paper an agent model is introduced that facilitates cooperation among agents. A multi-threaded multi-agent simulation framework is designed to test the model. The experimental results demonstrate that the model is significantly useful in achieving cooperation under various environmental constraints. It also allows agents to adjust their teammate selection strategies according to environmental constraints.*

## 1. Introduction

Typically, agents deployed in open environments may have different expertise and desires and act as self-interested entities to achieve their corresponding goals. In such systems, it is not possible to specify *a priori* the contexts in which an agent might need to interact with another for its service requirements. In addition, in a dynamic environment it is critical for agents to be able to adapt to changes in the environment by identifying the most influential aspects of multiple constraints.

This paper introduces a multi-dimensional model for partner selection based on attitudes and capabilities. In this model all agents are self-interested autonomous entities, and there is no central mechanism to control the system. The model allows autonomous agents to interact with each other and facilitates cooperation when required. The scenario studied in this paper is the following:

- Agents have different capabilities required for different tasks;
- The capabilities are designed in such a way that each agent is expert at only one capability, so cooperation of a team of robots is required to complete a task;
- Tasks are heterogeneous and have various requirements. In addition tasks have time stamps which indicate the expiry time of tasks.

In this scenario, agents are required to find a teammate and complete a task before time expires. Agents working on a task receive some reward if they perform the task before the time expires. The reward is based on the quality of work that the team has provided. Quality of work represents the degree to which the task is completed, which is proportional to the team capabilities undertaking the task. In addition to capabilities, agents have attitudes. The attitudes of an agent impact the behaviour of the agent for teammate selection [1]. Here, the behaviour of an agent towards teammate selection is referred to as the "decision strategy" of the agent.

A multi-threaded Java framework was designed to examine the proposed model. In this framework all the messages are asynchronous, and agents may accept or reject to help the requesting agent based on their attitudes. The framework deploys robotic agents in which each agent represents a robot. The simulation results demonstrate the effect of agents' attitudes and teammate selection strategies on rewards achieved by agents when cooperation is required. They also show that by using this multi-dimensional preference model agents can adapt to changes in the environment by adjusting their attitudes. Moreover, the effect of referrals on agent's performance is studied.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the environment and agent model. Section 4 details different strategies that agents may employ. Section 5 describes the simulation framework. Section 6 describes the learning mechanism. In Section 7 the experimental results are discussed, and section 8 concludes and outlines future work.

## 2. Related Work

Airiau *et. al*. [2] proposed a peer-to-peer approach to bring agents with complementary interests and

competences together. Agents team up to form self-sustaining teams of agents that can share their knowledge. Dutta *et. al*. [3] showed that by deploying a simple learning mechanism, self-interested agents with complementary expertises can learn to recognize cooperation possibilities and develop stable and mutually beneficial partnerships. However in these works the distribution of task types are uniform, and agents behaviors are fixed.

Sen *et. al*. [4] deployed a simple reinforcement learning mechanism in a large population of interacting agents. Their results show that simple reinforcement learning produces robust results relating to team formation. In their work favorable partnerships are predefined, and agents learn to recognize them. Ahn *et. al.* [5] introduced the concept of attitudes as a mechanism for choosing a partner. Attitudes define how an agent selects a partner by considering different situations. They also show how learned attitudes help in teammate selection.

The present work employs the concept of attitudes in a multi-agent scenario, in which each agent represents a mobile robot. The attitudes of an agent impact its teammate selection strategy. Here each robot is able to interact with only those robots which are within the same spatial region. In most previous work, either agents are able to interact with all other agents of the system, or all agents are provided with the same set of tasks and neighbors to provide uniform conditions for all agents [6]. However, in real multi-robot systems this assumption is usually not viable. In scenarios such as earthquakes or fires, there may be no possibility for global control over the number and distribution of tasks. The current work assumes a random distribution of tasks and agents to provide a realistic scenario in which the density of tasks and agents in different regions of the environment is variable.

Some other works assume that there is a pool of partner agents for the selection of teammates [5]. This assumption wastes some of the resources, since some agents are idle waiting most of the time. Our work assumes that all agents are active entities and can participate in either finding a task and partner or accepting an offer to become a teammate of another agent. We consider and emphasize the importance of the relative locations of the robots in connection with their partners as an additional dimension for selection of a teammate.

# 3. Environmental and Agent model

We simulated an environment where agents seek help from each other to complete a task. For simplicity it is assumed that a team is made up of two agents. However, the model could be easily extended to include more agents in a team. Each agent participating in performing a task is rewarded by a monitoring agent who oversees the simulation environment. The goal of each individual agent is to maximise its own reward.

## 3.1. Environment

The environment is divided into several spatial regions. A RFID tag is assumed to be deployed in each region and holds some information with respect to the geographical coordinates of the region and also the presence of tasks. The RFID tag of each region is called the environment tag. Agents are able to identify their own positions by reading the coordinate information of the nearest environment tag.

## 3.2. Agent Model

In this model each agent is equipped with a RFID tag and a RFID reader. Each agent is capable of communicating with other agents that are within the reading radio range of its RFID reader. The agent RFID tag holds the agent's identification number, position, and capabilities. The information on a RFID tag of an agent could be read by any other agent that has this agent within its visibility field. The position information enables agents to estimate their distance from their neighbors when selecting teammates. As agents move into an environment, they update their position information by reading the nearest environment RFID tag.

### 3.2.1. Capability

Robots have different capabilities that are useful in satisfying different task requirements. The capabilities of each robot are fixed and do not change over time. In this work two capabilities are assigned to each robot, capability *x* and capability *y*, and their values representing the quality level may range from 0 to 1**.**

- *Capability x* [0,1]: quality of capability *x* of robot.
- *Capability y* [0,1]: quality of capability *y* of robot.

### 3.2.2. Attitude

In addition to capabilities, each agent has attitudes that impact its decision-making. Social and behavioral psychologists sometimes describe attitudes as a predictor of behavior [1]. In that case attitudes can define the preferences of an agent for teammate

selection. In the current work each agent has the following attitudes:

- *att_nearness*: Attitude toward nearness: This refers to the agent's inclination to seek teammates that are physically close to the task.
- *att_quality*: Attitude toward quality: This refers to the agent's inclination to find teammates that provide high quality of service.
- *att_response time:* Attitude toward response time. This refers to agent's inclination to find a teammate as quickly as possible.

In this study each attitude is represented by a number between 0 and 1. For example, when an agent has a set of attitudes [0.2, 0.8, 0.0], it indicates that the agent has a low attitude toward nearness, a high attitude toward (predilection for) quality and no attitude toward response time. If an agent has a high attitude toward quality, then it prefers agents which will provide the highest quality with respect to the task requirements.

### 3.3. Task

Tasks are distributed in the environment and have different requirements that should be satisfied by the different capabilities of the robots. If there is a task in a region, then the corresponding environment RFID tag provides the task information. This information includes task position and task requirements. A task can be selected by an agent if no other agent is working on it. In this work we assume that each task has a set of two requirements $r_x$ and $r_y$ in which we have:

- $r_x$ [0,1.5]: quality required for *x* dimension of task.
- $r_y$ [0,1.5]: quality required for *y* dimension of task.

The requirements range is somewhat arbitrarily chosen to range between 0 and 1.5. Therefore, agents are required to cooperate in order to satisfy their requirements. $r_x$ and $r_y$ are satisfied by capabilities *x* and *y* of an agent respectively.

### 3.4. Reward Mechanism

Each task requires agents with certain capabilities and has an associated reward. The reward is distributed equally to the agents that participated in completing a task within a specified time constraint. When agent $A_i$ works on a task *t* with *n* number of agents as a team and the team completes the tasks, then each agent participating in completing the task receives the following reward:

$$R_{A_i}(t) = \frac{R(t)}{n}$$

*R(t)* is the original reward associated with each task.

If the team can satisfy only some part of the task because the capabilities of all the participating agents in completing the task is less than the task requirement, then the reward that each agent $A_i$ receives is defined as follow:

$$R_{A_i}(t) = \left(\frac{\sum_{k=1}^{n}(A_k)_x}{r_x} + \frac{\sum_{k=1}^{n}(A_k)_y}{r_y}\right) * \frac{R(t)}{n}$$

*Where*

$(A_k)_x$ ,$(A_k)_y$: Capability x and *y* provided by $k^{th}$ team member

$r_x$ ,$r_y$: *x* and *y* requirements of task

*R(t):* Total reward for completing task t

*n*: the number of agents in the team

For example assume that two agents complete a task which offers the total reward of 7. Then the amount of reward that each agent receives is 3.5. Accordingly, if two agents just finish 80% of that task, then each agent receives a reward of 2.8.

## 4. Agent decision strategies

In order to select a teammate, each agent ranks all its available neighbors based on their expected performance and selects the best one. We examine four primary strategies that an agent can employ for selecting a partner. The decision strategy of each agent depends on the attitudes of the agent. Table 1 shows the different attitudes of agents and their corresponding decision strategy.

| Attitudes *(nearness,quality,response_time)* | strategy |
|---|---|
| (0,high,0) | *Best_possible* |
| (low,high,0) | *Best_available* |
| (high,0,0) | *Nearest_available* |
| (0,0,high) | *Impatient* |

### 4.1. *Best_possible* teammate strategy

An agent that employs this strategy only selects another agent as its teammate if the two agents as a team can complete the task. It is formally represented as:

$$A_{ix} + A_{jx} \geq r_x \quad and \quad A_{iy} + A_{jy} \geq r_y$$

Where $A_{ix}$, $A_{iy}$ and $A_{jx}$, $A_{jy}$ are the *x* and *y* capabilities of agent $A_i$ and $A_j$ respectively. $r_x$ and $r_y$ refer to the *x* and *y* requirements of a task respectively. The agent ranks its neighboring agents that satisfy the above conditions based on their usefulness value and selects the top

ranked agent as its teammate. The helpfulness value for each agent is calculated based on the agent's distance to the task.

$$H_{Aj} = dist_{Aj}$$

In this equation $dist_{Aj}$ is the distance rating of partner agent $A_j$ ($0 \leq dist \leq 1$).

#### 4.1.1. Referral

An agent with *best_possible* strategy does not select a partner unless the two agents as a team could complete the task. An agent with this strategy may simply fail to achieve some reward due to not having an agent with complementary capabilities within its visibility field. So the referral mechanism could be employed to improve the performance of agents with *best_possible* strategy.

Referral systems are multi-agent systems whose members may follow a cooperative protocol by providing referrals to another agent, thus sharing their knowledge about service providers and enabling improved service selection. In current work, when an agent does not have the required capabilities then the agent checks its visibility range and if it could see any agent with the required capabilities then it sends a referral to the requested agent. An agent refers to another agent if the agent itself does not have the required capabilities and task time has not expired. If there is less than half the required time left to solve the task, then the agent dose not send a referral and removes the request from its request queue.

### 4.2 *Best_available* teammate strategy

Agents that employ this strategy gain some rewards by partially completing tasks. These agents may select teammates that only have some of the expertise required by the task, and they receive a partial reward for the part of the task that is completed. For each available agent in the neighborhood, the quality of the capabilities is calculated based on the following equation in which Q is the quality of capabilities of agents $A_i$ and $A_j$.

$$Q = r_x * (A_{ix} + A_{jx}) + r_y * (A_{iy} + A_{jy})$$

$A_{ix}$, $A_{iy}$, $A_{jx}$, and $A_{jy}$ are the values for $x$ and $y$ capabilities of agents $A_i$ and $A_j$ respectively and $r_x$ and $r_y$ refer to $x$ and $y$ requirements of the task. The total $x$ capabilities of the agents are paired with $x$-requirement of the task, and the total $y$ capabilities of agents is paired with $y$-requirement of the task. So if for example the task has a high $r_x$ then, this equation weighs capability $x$ of agents more heavily.

For teammate selection each agent considers all available local agents and ranks them based on their helpfulness value. Given the quality of capabilities, the agent calculates the helpfulness of each possible partner as follows:

$$H_{A_j}^{A_i} = Q * A_{i_{att\_quality}} + dist_{A_j} * A_{i_{att\_nearness}}$$

In this equation $A_{i\ \text{att-quality}}$ and $A_{i\ \text{att-nearness}}$ are the attitude of agent $A_i$ toward quality and nearness respectively. The agent's attitude toward quality is associated with the quality that the agent can provide with respect to task requirements, and an agent's attitude toward nearness is associated with the distance rating of the potential partner. According to this equation, an agent that uses this *Best_available* strategy weighs quality of work more than distance. An agent selects a teammate that has the highest value of $H$.

### 4.3 *Nearest_available* strategy

An agent that employs this strategy selects an agent that has the least distance to the task and receives a partial reward for completing some parts of a task. Agents with this strategy receive partial rewards proportional to the completed part of the task.

### 4.4 *Impatient* teammate strategy

Agents that employ this strategy do not wait to receive responses from all the requested agents, but select the first agent that responds to their requests as their teammate.

## 5. Framework description

A multi-agent framework was designed to test the proposed model. The framework was implemented using Java language, with each agent having its own thread of execution and all the messages of the system being asynchronous. The concurrency feature of the system provides a better testing environment for real agent applications. The framework allows agents to run multiple auctions over various tasks concurrently.

In this system each agent has two queues: a request queue and a response queue. The request queue stores the requests from other agents and the response queue stores the positive responses of other agents. Initially each agent observes its environment and stores the task's and agent's information in its memory. If there is a task within its visibility field, then the agent selects the task and changes the status of the task to *unavailable* for other agents. In cases where there are

several tasks, each agent selects a task that has the least distance to the agent. After finding a task, agents start looking for teammates. In order to find a teammate, the agent sends a request message to all available agents in its neighborhood and waits for a certain amount of time to receive their responses (note that agents with *Impatient* strategy do not wait). Since tasks have time stamps, agents are required to send their requests to their neighbors in the order of their helpfulness. This is to make sure that the agents who have higher helpfulness values will receive the request before others and therefore they have enough time to respond. This is especially critical when time constraints of tasks are very tight and the requesting agent waits for a short time.

Algorithm 1 Agents partner selection mechanism

```
1.  Observe; the information of available agents and tasks
2.  t=Find a task
3.  If (t!=null){
        Send a request; to all available neighbors
        Wait (time); if applicable
4.  While (agent.teammate=null AND time≤(task time/2)){
        selectedResponse=Select the best response
        selectedRequest=Select the best request
        if (selectedResponse.teammate!=null){
            If (H(selectedResponse)≥H(selectedRequest)
                Team up; with selectedResponse
            else{
                send response to the selectedRequest
                requestQueue.remove(selectedRequest)
                wait(time)
            }}
        else
            responseQueue.remove(selectedResponse)
        }
        responseQueue.clear
        requestQueue.clear
    }
5.  if(t!=null AND task time expired)
        drop t
6.  If(t!=null AND teammate!=null)
        moveTowardTask
    else
        moveRandomly
```

When the waiting time is over, then the agent starts processing the received responses and requests. The agent ranks its response queue and request queue based on the helpfulness value of agents. If there is a request with a higher helpfulness value than all the responses, then the agent sends a response to the requesting agent. If the requesting agent does not select this agent within a short time, then the agent repeats the explained process (while loop in the above algorithm) until it finds a teammate or time expires (in this framework agents spend 1/3 of the task time on finding a teammate and the rest for moving toward a task). An agent only considers a request if more than half the time required by the task is left, otherwise it removes that request from its request queue. The algorithm 1 shows the pseudo-code which each agent runs constantly.

Although this is a concurrent system, the CPU runs the threads sequentially and in a random order. Moreover, the amount of time that is spent on executing each thread is different. In order to make sure that all threads are approximately run for the same amount of time by the CPU, all the threads were set with a high priority. In addition, at the end of each simulation run, a thread waits for 50 milliseconds. This puts the current thread to sleep and allows the CPU to run the next thread. This is especially important when the time constraints of tasks are very tight and the response should be given quickly.

## 6. Learning attitude toward time and quality

In dynamic environments agents are required to adapt to new conditions in order to maximize their rewards. In this work agents can adapt to new conditions by changing their attitudes. Agents change their attitudes based on the feedback that they receive from the environment. For instance there might be some conditions when agents receive a low reward because time is very tight. In this situation agents should learn to decrease their attitude toward quality and increase their attitude toward nearness. A simple reinforcement learning mechanism is employed to alter the agent's attitudes. To demonstrate the effect of learning, agents with *best_available* strategy are deployed.

An agent performs with its current set of attitudes for a certain amount of time and stores a copy of the attitudes and the total reward received. Then the agent increases its attitude toward one dimension and decreases values in the other dimensions. Then the agent performs for the same amount of time with its new set of attitudes. If the reward that the agent receives is more than the reward gained by its previous set of attitudes, then it continues to change its attitudes the same way; otherwise, it increases its attitude toward the other dimension. The following reinforcement learning formula is used to update the agent's attitudes:

$$a_{t+1}=a_t+\beta(R(a_{t+1})-R(a_t))$$

In this equation $a_{t+1}$ is the new value for attitude. $a_t$ is the previous attitude of the agent. $R(a_{t+1})$ is the actual reward received for new attitudes and $R(a_t)$ is the actual reward received by previous attitudes. $\beta$ is the learning rate.

## 7. Experiments

A series of experiments was conducted to study the performance of the proposed model. The simulation environment is a grid of 100 by 100 cells in which each cell refers to one square of the grid. There are 120 robots with different capabilities. Robots can only move vertically and horizontally one cell at a time. There are 1000 tasks with different requirements placed randomly in the environment. The reward for each task is a fixed number (7). All times are in milliseconds.

### 7.1. The effect of time on agents' reward

In this experiment there are four groups of agents in which each group employs one strategy (*best_available*, *best_possible*, *nearest available* and *impatient*). Four different runs of simulation were run under various time constraints, where in each run, agents with one strategy were deployed. The total reward achieved by each group was measured under various time constraints of 500, 1000, 2000 and 3000 milliseconds.

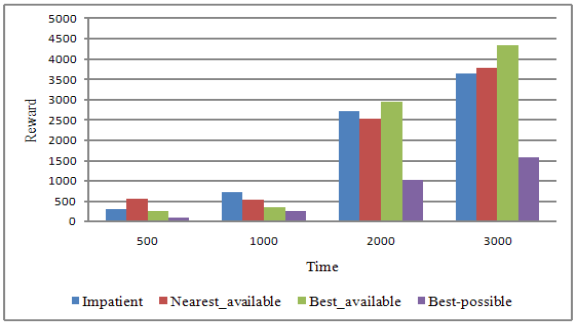


**Figure. 1 Effect of agent's strategy on agents reward.**

Figure 1 shows that when time is very tight (500), agents which employ *nearest-available* strategy outperform the other groups. In this situation, all groups perform somewhat impatiently due to lack of enough time. Therefore, agents with *nearest_available* strategy who select the closest available teammate perform better than other groups. This is due to the fact that agents have enough time to move toward their task and work on it. However, other groups of agents may select a teammate that might be further away and therefore the team may fail to reach the task position before time expires. When time is a bit more relaxed but still relatively tight (1000), then *impatient* strategy outperforms other strategies. An Impatient agent selects the first responding agent as its teammate and does not wait for others to respond. This increases their chance of completing a task before expiry time. When time is more relaxed (1000 and 2000) *best_available* strategy outperforms the other strategies. This is the result of selecting high-quality teammates. The agents with *best_possible* strategy perform worse under various time constraints. This is the effect of perfectionist attitudes of these agents. This approach is useful when there is more incentive in completing a job (refer to next experiment).

### 7.2. The effect of referral on performance of agents with *best_possible* strategy

In some situations the tasks are required to be done completely. For instance, if the task is to clean a minefield, then partially completing tasks is not appropriate. Therefore for certain scenarios agents are required to complete the tasks as opposed to partial completion. In order to improve the performance of agents with *best_possible* strategy a referral mechanism is deployed.

Three runs of simulation were run. In the first run (no. of hops=1) no referral was deployed. In the second run (no. of hops=2) if the requested agent does not have the required capabilities but it could see an available agent with the required capabilities, then it sends a referral to the requesting agent.

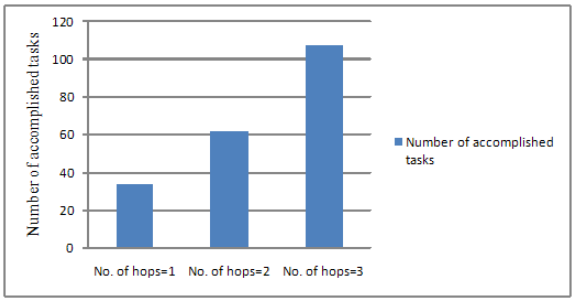


**Figure. 2 Effect of referral on agents' performance.**

In the third run (no. of hops=3), the requested agent may ask its neighbors whether they could see an agent with the required capabilities. The time constraint was set to be very relaxed, so agents have enough time to send and process referrals and the reading range of the agents was 5. Figure 2 demonstrates that when agents deploy referrals (no. of hops=2 and no. of hops=3) their performances improve.

### 7.3. The effect of learning and adaptation

The aim of this experiment is to show how agents can adapt to dynamic changes in the environment. In this experiment agents with the *best_available* strategy are deployed, and the time constraint of tasks is set to a

low value of 500 milliseconds. Two runs of simulation were performed. In the first run agents do not learn, and in the second run learning is employed. Since time is very tight, agents with *best_available* strategy eventually change their attitudes by using the learning mechanism described previously. So they increase their attitude towards nearness and decrease their attitude towards quality. By changing the attitudes agents eventually adapt *nearest_available* strategy.

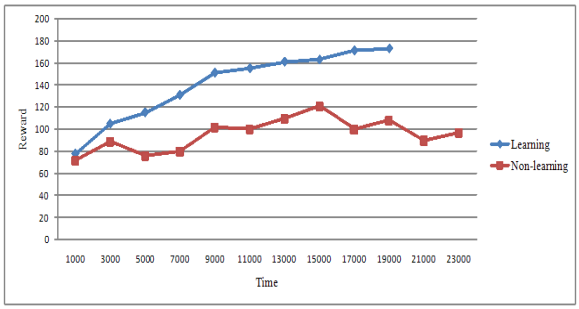


**Figure. 3 Reward of agents within each period of time.**

Figure 3 shows the total reward achieved in each period of time (every 1000 milliseconds). It shows that the learning mechanism improved the reward of agents and decreased the total time required to complete the tasks.

## 8. Conclusion

This paper has presented a multi dimensional model for partner selection based on capabilities and attitudes. A multi-threaded Java framework was designed to test the proposed model. The designed framework provided a realistic environment for testing the model. In this model each agent may accept a request to provide a resource or reject it based on the agent's attitude and strategy. All messages are asynchronous, thus agents do not wait for the response after the waiting time expires. We have evaluated the model by using a grid type simulation environment. The result shows that *nearest_available* strategy is useful when time constraints of tasks are extremely tight. *Impatient* strategy is suitable for situations when time is relatively tight. *best_available* strategy is preferable in situations when time constraints of tasks are more relaxed and there is enough time for agents to find a high quality teammate. *Best_possible* strategy is preferable in situations where tasks are required to be fully accomplished as opposed to be performed partially. The experiments demonstrated that, by using the referral mechanism, an agent's performance in terms of the number of completed tasks was improved. The referral mechanism allows agents to find their required resources if there is any agent with the required expertise close to the location of the neighboring agents. We kept the number of hops that a message can be transferred relatively low to keep the agents' information local. We demonstrated that referrals improved the performance of agents but under very relaxed time constraints. Note that referrals are probably not appropriate for emergency situations. It was also shown that, by using a simple reinforcement learning mechanism, agents had the ability to adapt their partner selection strategy to the changes of the environment. The result indicated that adapting attitudes could significantly improve the performance of the agents.

An interesting approach for future work could be considering different rewards for various tasks depending on the size of the tasks. In addition a cost could be associated with giving a referral. This is interesting, since it may hamper the agent's self-interest. Moreover, it is valuable to study whether agents of each group may have a preference for agents of other groups (strategies). Another interesting aspect could be studying the effect of deceptive referrals where only the referrals of trustworthy agents should be considered.

## 9. References


[1] Tesser, A. and Shaffer, D., *Attitudes and Attitude Change*. Annual Review of Psychology, 1990. **41**(1): p. 479-523.

[2] Airiau, P., Sen, S. and Dasgupta, P., *Effect of joining decisions on peer clusters*, in *Proceedings of the fifth international joint conference on Autonomous agents and Multiagent systems*. 2006, ACM: Hakodate, Japan. p. 609-615.

[3] Dutta, P.S., Moreau, L. and Jennings, N.R. *Finding interaction partners using cognition-based decision strategies*. in *Proceedings of the IJCAI-2003 workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*. 2003.

[4] Sen, S., Gursel, A. and Airiau, S. *Learning to identify beneficial partners*. in *the Proceedings of the Workshop on Adaptive and Learning Agents at the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. 2007. USA.

[5] Ahn, J., DeAngelis, D. and Barber, S. *Attitude Driven Team Formation using Multi-Dimensional Trust*. in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*. 2007.

[6] Dutta, P.S. and Sen, S., *Forming stable partnerships*. Cognitive Systems Research, 2003. **4**(3): p. 211-221.

[7] Banaei-Kashani, F. and Shahabi, C. *Criticality-based analysis and design of unstructured peer-to-peer networks as" Complex systems*. in *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*. 2003.