COOPERATIVE CONTROL OF A MULTI-TIER MULTI-AGENT ROBOTIC SYSTEM FOR PLANETARY EXPLORATION

P. Isarabhakdee & Y. Gao*

*Corresponding author

Address: Surrey Space Center, University of Surrey, Guildford, Surrey GU2 7XH, UK
Email yang.gao@surrey.ac.uk; Telephone: +44 1483 683446

ABSTRACT

The aim of this paper is to address some key design issues of a multi-tier multi-agent system (M2S), such as to define the *control architecture* of the multiple agents within the two tiers (e.g. centralized and distributed), and the *control or reasoning algorithm* that allows individual agent to organize its behaviors, learn and/or adapt to unknown and dynamic conditions.

1. INTRODUCTION

Recently, a multi-tier multi-agent system including unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) has been recommended as a promising approach for future planetary reconnaissance missions [21]. This approach is primarily based on cooperative control between the UGV and UAV. For example, multiple UGVs with feedback from a UAV can potentially survey a large planetary area more quickly and safely and thus improve the scientific return of the mission. The M2S has homogeneous agents in each tier and thus improves redundancy in the mission. It can also be extended to 3 or more tiers of heterogeneous agents, e.g. by including satellites in orbit as the 3rd tier agents.

The M2S can be configured in many different ways in terms of communication (data sharing) and cooperative control among the agents. In this paper, we propose a M2S with some basic configuration as follows. This results in a simple but effective system that can be implemented in a planetary mission at relatively low cost.

- Each UAV or UGV is a fully functional agent and can work independently.
- Each UGV receives data generated by the UAV periodically as an additional sensor input, such as terrain condition, localization data, etc.
- Each UGV reports to the UAV periodically on its health condition.
- The UGV can send emergency signal to the UAV if there is a fault in its system. The UAV decides whether to relay the UGV data to other UGVs and to whom. This means the UGV can receive data generated by other UGVs via the UAV and there is no direct communication between the UGVs.

Under this basic configuration, the UAV acts more like a coordinator of information sharing. If the UAV fails to work properly, the UGVs are still capable of making decisions based on their onboard sensor data. Complete control architecture of the M2S also needs to explain the control configuration within each agent. This is normally associated with design of an automated control algorithm for the agent operation.
Over the half century, NASA has been developing some single-tier multi-agent control architectures for planetary applications. One example is the *Control Architecture for Multi-robot Planetary OUTpost* (CAMPOUT) [12]. The CAMPOUT focuses on coordination of the agent behaviors. The *primitive* behaviors are basic behaviors of each agent that can form the so-called *composite* and *group* behaviors. The *shadow* behaviors result from communication between the agents and can be used to form group behaviors. A behavior coordinator is responsible to handle conflicts between current behaviors and feedback from actuators by setting up behavior priority. Command fusion is the key for this architecture, which regulates group behaviors of each agent to be cooperative, not competitive. The CAMPOUT architecture works statically with existing (or pre-defined) behaviors of the agents. This

implies its ability to adapt and cope with an uncertain environment is limited.

The *Multi-rover Integrated Science Understanding System* (MISUS) is another example of a multi-agent control architecture designed for planetary exploration [20]. The general configuration is considerably extended from the CAMPOUT. The key features added to the architecture include data analysis via learning and dynamic planning/scheduling components. The rovers or UGVs can therefore reason how to achieve a goal, and learn to evaluate what new goal should be performed such as observing and gathering scientific data. The agent behaviors are selected dynamically based on CASPER planning and scheduling [17]. A central planner generates the team objective and assigns it to each agent. Each agent will inform status (i.e., whether or not the team goal has been achieved) back to the central planner for re-planning. The MISUS enables the agents to reason where they learn cooperatively how to gather and select their behaviors to achieve the team goal. The MISUS cannot though be used to control a M2S directly.

The *ALLIANCE* proposed by Parker in [9] is architecture for fault tolerant multi-agent cooperation on mobile robots. By using adaptive behavior selection, it allows agents to adapt to high-risk fault and dynamic environment. The architecture requires the agent to be capability of communicating with other agents and recognizing the behavior performed by other agent in case of communication failure. This is very costly to implement in a planetary reconnaissance mission and thus is not compatible with the basic configuration of the proposed M2S.

In this paper, we propose a novel *Reasoning and Control Architecture* (RACA) that enables autonomous operation of the M2S. The remaining of the paper introduces the working principle of the RACA and associated learning technique. Computer simulation can validate results are presented at the end of the paper followed by performance evaluation both individually and as a team. Multi-agent learning basically allows agents to collaboratively select actions that maximize their own, as well as the team output, and to cope with uncertainties and unpredictable failures.

2. PROPOSED RACA

The RACA is proposed to enable autonomous operation of the M2S and satisfy planetary reconnaissance mission requirements. Main design objectives are summarized as follows:

- To support autonomous cooperation between the UAV and UGV of the defined M2S;
- To enable decision making of the agent via learning in the unknown environment and select effective behaviors to fulfill both individual and team goals, given a mission task;
- Is capable of fault tolerance for the M2S;
- Is scalable to more complex M2S configurations.

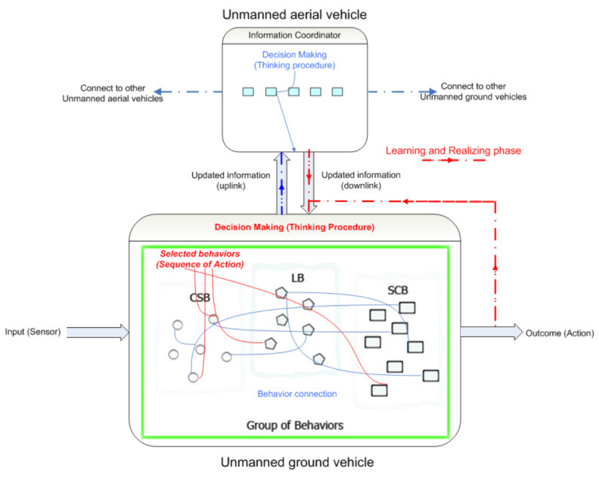
Unmanned aerial vehicle
Information Coordinator
Decision Making (Thinking procedure)
Connect to other Unmanned aerial vehicles
Connect to other Unmanned ground vehicles
Learning and Realizing phase
Updated information (uplink)
Updated information (downlink)
Decision Making (Thinking Procedure)
Selected behaviors (Sequence of Action)
LB
CSB
SCB
Input (Sensor)
Outcome (Action)
Behavior connection
Group of Behaviors
Unmanned ground vehicle

Fig 1 RACA Basic Configuration

The RACA is illustrated in Fig 1 that explains the basic configuration of the M2S using one representative UAV and UGV. Each agent either UAV or UGV is an intelligent and behavior-based entity. The architecture allows information exchange between the UAV and UGV on a regular basis as well as in an emergency condition. Such exchange can be considered as additional sensor input to each agent and used for selection of its behaviors. The RACA includes the control configuration and learning algorithms to enable the agents make decisions and respond effectively to a mission task. The agent can learn to choose the optimal set of behaviors for a given individual and team goal. Fault a UGV (red circled in Fig2) is running out of power and informs the UAV. The decision of the UAV is to tolerance of the UGVs is realized by the UAV coordination.

2.1 Information Coordinator

As aforementioned, the UAV is an information coordinator in the M2S. There is no direct communication between the UGVs. Data is updated between the UAV and UGV periodically. A major function of the UAV is to screen data from the UGV and relay to other UGVs based on certain decision criteria (or goal). This helps to decrease computation and communication load of individual UGV and still allows cooperation between the UGVs. For example, relay this information to a selected number of UGVs (green circled) that are in better position to help the troubled UGV (such as in close vicinity, and finishing own tasks, etc).

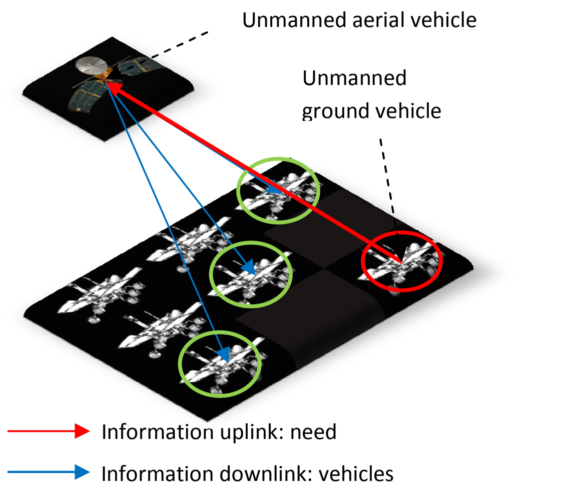

Fig 2 UAV Decision making

2.2 Behavior Definition and Representation

In the RACA, the agent behaviors are classified into three groups:

- **Common Sense Behaviors (CSB or Skills)** represent a set of basic behaviors or *skills* of an agent before any learning takes place, e.g. moving from point to point, hazard detection for UGV, and relay data for UAV, etc.
- **Learned Behaviors (LB or Roles)** represent behaviors learnt by the agent in a specific state or *roles* the agent play to achieve the individual goal.
- **Specifically Cooperative Behaviors (SCB or Group)** represents behaviors post to learning of an agent in a specific state according to the team goal or strategy.

The learning process to select LB and SCB is carried out in sequence in order to achieve individual goal and team goal respectively. The LB are obtained from the CSB after individual learning. Selected LB are the most effective CSB to achieve the individual goal in a specific state. Similarly, SCB learnt from the CSB and/or LB are used to achieve the team goal. If to use the same example as in the **Error! Reference source not found.**, Fig 3 shows the decision making process of a green circled UGV. Similar theory applies to the UAV, in which case cooperative learning is triggered by fault signal received from the UGV.

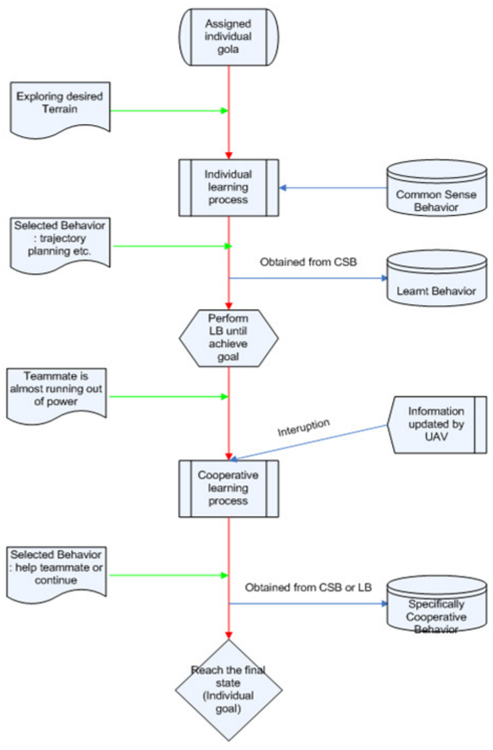
Assigned individual gola
Exploring desired Terrain
Individual learning process
Common Sense Behavior
Selected Behavior : trajectory planning etc.
Obtained from CSB
Learnt Behavior
Perform LB until achieve goal
Teammate is almost running out of power
Interuption
Information updated by UAV
Cooperative learning process
Selected Behavior : help teammate or continue
Obtained from CSB or LB
Specifically Cooperative Behavior
Reach the final state (Individual goal)

Figure 3 Decision Making Process

2.3 Learning and Behavior selection mechanism

In the RACA, *reinforcement learning* (RL) technique is employed for the agent to select behaviors automatically. The RL has been a popular approach to support cooperative control of the multi-agent system [2-4, 8, 11] as this technique does not require off-line training nor rely on a 'teacher' knowing the absolute answer to the question. It enables the agent to learn by interaction with the unknown environment and learn from 'experience' by maximizing a learning reward associated with its goal. The RL results in a pair of behavior and state (as a sequence) for the agent to achieve particular objectives.

At the beginning, each UGV is assigned its individual goal before the Q-learning procedure. The UGV then learns to obtain a sequence of effective behaviors (i.e. LB) to reach the goal, known as *Individual learner* (IL) process. In case the agent is interrupted by UAV due to updated information, the agent learns for the SCB to optimize both individual and team goals, known as *Joint action learner* (JAL) process. Basic formulation of RL and Q-learning has been developed widely and variously, however many derivatives are still based on RL basic formulation as fundamental terminologies are proposed below

- t - discrete time step
- s_t- state space at time t
- a_t- action performed at time f, related to s_t
- r_t - reward at time t, related to s_{t-1} *and* a_{t-1}
- π - policy, mapping state to action
- π^* - optimal policy (design goal)
- v – value function, by $v^{\pi}(s)$ is value of state s under policy π

Finding optimal behavior through particular models is another step to be concerned about. Generally, the model can be divided appropriately into finite and infinite horizontal models. But to satisfy mathematical convergence (bounding) without information of the exact length of learning life time, infinite horizontal model is thus preferred, as mentioned below.

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \qquad (1)$$

The above equation is a general form of infinite horizontal model with discount rate (γ^t). Learning is efficient in a long run as it is bounded by γ^t to ensure convergence. The performance of learning is measured typically that learning algorithm converses to an optimal point and speed of convergence is satisfied seriously. For the most general problem, *delayed reward* is a form usually described in RL. Not only is immediate reward determined, but also considers future reward obtained at the next state. In association with this scenario, problem is modeled by using Markov Decision Process [14][6] which is described as follows.

- A set of Action : A
- A set of State : S
- Scalar Reward : $R : S \times A \rightarrow \Re$
- State Transition Function : $T(s, a, s')$, mean probability that performing action a at state s leads to state s'

The proposal of this process is to find Optimal Policy [14]. Value function plays an important role in this mathematical formulation. In order to obtain π^* policy, maximum value function can be derived by expectation of infinite horizontal reward model as the following.

$$V^*(s) = \max_{\pi} E\,(\sum_{t=0}^{\infty} \gamma^t r_t) \qquad (2)$$

A more practical form would be:

$$V^*(s) = \max_{a} \,(\, R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')\, V^*(s')) \,, \forall s \in S \qquad (3)$$

*By equation (3) optimal policy can be also inversely specified.

Since solving the equation directly to find optimal policy is quite complicated, the learning process employs exploration method which has two alternative ways to proceed, Model-free and Model based. For Model – free, which is a general form, Q – learning is simpler understanding and modeling than that of Model based. Considering--

$$V^*(s) = \max_a Q^*(s,a) \qquad (4)$$

Equation (3), after replaced by equation (4), becomes

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_a Q^*(s',a')$$

$$\forall s \in S \qquad (5)$$

Q value can be accurately estimated by updating (learning) recursively. Algorithm function can be called value iteration method [6][14], solved by dynamic programming. Action with maximum Q value is selected to perform in each state following this formulation.

$$Q(s,a) := Q(s,a) + \propto \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \qquad (6)$$

The approach to reach optimal state is highly dependent on how the agent acts whilst experiencing the environment, as well as having sufficient trial process in each state-action pairs. To ensure that Q learning would be converged to certain value, exploration strategy, therefore, must be well-defined. Conclusively, the procedure of learning algorithm yields the following:

```
$Initialize\ Q(\boldsymbol{s},\boldsymbol{a})\ arbitrarily$
$Repeat\ (for\ each\ episode):$
    $Initialize\ \boldsymbol{s}$
    $Repeat\ (for\ each\ step\ of\ episode): - followed\ by\ Exploration\ and\ Exploitation\ method$
    $Choose\ \boldsymbol{a}\ from\ s\ using\ policy\ derived\ from\ Q$
    $perform\ action\ \boldsymbol{a}, observe\ \boldsymbol{r}, \boldsymbol{s}'$
    $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a}) <= Q(\boldsymbol{s},\boldsymbol{a}) + \propto (\boldsymbol{r} + \boldsymbol{\gamma} \max_{a'} \boldsymbol{Q}(\boldsymbol{s}',\boldsymbol{a}') - \boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$
            $\boldsymbol{s} <= \boldsymbol{s}'$
    $untill\ \boldsymbol{s}\ is\ terminated\ or\ complete\ length\ of\ \boldsymbol{s}$
```

After completing the above procedure, behavior with maximum Q value in particular state has a priority to perform. Keys of Q learning are firstly how to model reward function effectively in both IL and JAL, secondly how to select suitable exploration and exploitation method. Some of related researches show Q learning issue problem is that it's slowly about to converge because effective exploration approach is not designed.

2.3.1 **Individual Leaner (IL)**

This learning scenario is almost similar to a general form of RL mentioned before. Algorithm follows classic approaches but is only partly adjusted as appropriate. Preliminarily, each UGV has the CSB, and is then assigned a reward function which only happens at the final state (generally where goal exists). Learning process is executed until it obtains the sequence of behaviors to achieve individual goal. All selected behaviors are mapped into LB. In IL approach a key to ensure that this algorithm can reach high performance is *Weight factor*s. They are very critical in forcing algorithm to convert optimally. Weight factors can be listed as:-

- *Discount rate factor*: Used to decide weighting between an immediate reward and future reward, ranging from 0 – 1. The lower weighting is, the higher UGV pay attention on immediate reward.
- *Learning rate factor*: Adjust learning levels whether to keep high learning rate or de…..crease leaning rate, ranging from 0 – 1 (maximum learning).
- *Exploitation factor*: While in learning process, selection of each behavior is biased based on past and current probability value. For example, a behavior with higher Q value in current state is

more likely to be selected. In our algorithm, *Boltzmann exploration* is deployed to define probability of each behavior, as can be seen in the following equation:-

$$\frac{e^{Q(b)/T}}{\sum_{b'} e^{Q(b')/T}} \qquad (7)$$

Q(b) is Q value of behavior b
T is temperature parameter which will control the convergence rate, and which will decrease over time.

Whether or not the learning process is terminated depends on two main conditions. Termination happens when learning rate factor is equal to zero, which means no learning continues, or when the Q value is conversed to optimal value which means there is no change in decision while learning is continued.

2.3.2 **Joint Action Learner (JAL)**

In achieving team goal, each UGV is aware of other UGV's action in a dynamic environment. As previously mentioned, it begins cooperative learning since the information has been updated via UAV, informing that there is an unexpected event or a disturbed strategy. The major difference between IL and JAL is the value estimation of behaviors. Rather than estimating the Q value of each behavior, JAL algorithm examines *Reduced Profile* which contains a set of behaviors potentially performed by each UGV in specific state. Each selected behavior will then be regarded as SCB of each UGV. The goal is the UGV tries to select behaviors to maximize Q value of reduced list. For example, at state S, UGV A is going to perform behavior A0 at the beginning. However, after observing that UGV B is performing behavior B1, Q value is therefore assigned to the profile which is a joint behavior <A0, B1>. After UGV A learns sufficiently to trace the optimal behavior regarding UGV B, it can switch to the behavior maximizing Q value of reduced profile to satisfy cooperative condition. To implement this idea, designed algorithm is introduced as the following:-

```
$Begin\ initial\ state\ \boldsymbol{s}$
$for\ involving\ agent\ \boldsymbol{i}$
$-\ agent\ that\ recieve\ information\ from\ UAV$
        $Initialize\ Q(\boldsymbol{s},\boldsymbol{p_i})\ based\ on\ IL\ strategy$
$Repeat\ (for\ each\ episode):$
        $select\ profile\ \boldsymbol{p_i}$
$-followed\ by\ Boltzmann\ Exploration\ method$
        $Repeat\ (for\ each\ step\ of\ episode)$
            $perform\ \boldsymbol{p_i},\ observe\ \boldsymbol{r},\boldsymbol{s}',\alpha_i\ \leq$
  $learning\ rate\ of\ \boldsymbol{p_i}\ -$
$-weighted\ total\ learning\ rate\ of\ all\ profile$
            $\boldsymbol{Q(s,p_i)}\ <=\ Q(\boldsymbol{s},\boldsymbol{p_i}) + \propto_i (\boldsymbol{r} +$
$\boldsymbol{\gamma}\ \mathbf{max}_{\boldsymbol{a}'}\ \boldsymbol{Q}(\boldsymbol{s}',\boldsymbol{p}_i') -\ \boldsymbol{Q(s,p_i)}$
$learning\ rate\ of\ \boldsymbol{p_i} = \frac{(learning\ count+1)}{current\ learning\ length}$
            $\boldsymbol{s}\ <=\ \boldsymbol{s}'$
  $untill\ \boldsymbol{s}\ is\ terminated\ or\ complete\ length\ of\ \boldsymbol{s}$
```

Regarding to communication scheme, UGV is interrupted by updated information and start learning to act cooperatively. Only the involved UGVs take part in this cooperative strategy, displayed in Fig 2. At first UGV initially believes in selected behavior from LB. Afterward, it tries to replace other possible behaviors into the same reduced profile. Based on the exploration strategy, the reduced profile which has the highest Q value is most likely to be selected. In each cycle, the selected reduced profile increases the learning rate which means it has higher possibility to converse eventually and it is going likely to be the optimal profile. The loop is terminated when one of involved UGVs decides to follow its selected behavior in reduced profile as selection probability is above threshold. Meanwhile other UGVs continue IL strategy based on LB.

3. SIMULATION

To demonstrate that RACA works effectively with space reconnaissance mission, the simulation is developed so as to reflect how RACA plays critically in achieving mission goal. In general, it consists of control systems with learning capabilities embedded, and also the modules representing the world model and the role of UAV, where information coordination of each UGV is happened**.**

The simulation focuses mainly on comparing performance between single UGV and multi UGVs with UAV in this particular scenario. For a single UGV, UGV is assigned to survey an unknown terrain by using information only from itself (local vision and sensor). Additionally, the single UGV has no capability of learning before making a decision. In other words, the UGV can only perform behaviors commanded by mission controller at ground station. Surveying process continues until UGV receives stopping signal from the mission controller. However, the single UGV's qualification is compensated in operating at a very high speed, and having very advanced equipment to achieve the mission. On contrary to the single UGV, multi UGVs work cooperatively with UAV by exchanging real time information. They also have a learning capability to help selecting an optimal behavior in a particular condition (state). Nevertheless, capability of each UGV is scaled down as the need of constructing 5 UGVs and 1 UAV.

For an evaluation, the team of UGVs is capable of surveying all terrains. This mission takes explicitly lower time than the single one, including investigating and data collecting processes, as well as failure analyzing and repairing. Furthermore, the mission is very successful in collecting essential data from terrains and found objects. By communicating via UAV, failures happening during the mission are recovered quickly and effectively, demonstrating high redundancy. This explicitly means the increase of mission reliability. In view of learning capability, this allows UGVs to select optimal behaviors and to coordinate an action with other UGVs appropriately. It can be implied that this ability helps UGV to eliminate unnecessary actions for the early time. Comparatively, although single UGA perform surveying terrain faster than team of UGVs, team of UGVs achieves mission objectives in all aspects. Finally the result of simulation is illustrated by comparing single agent and multi agents in Table 1.

Table 1 Comparison for Single UGV and Multiple UGV with UAV performance

	Single UGV	Multiple UGVs with UAV
Time Consuming	Consume more time	Consume less time
Information Gathering	Gather only terrain information	Gather successful any relevant information on the terrain
Reliability	If failed, it means mission is no longer work	If failed, UGV is analyzed and repaired by one of other UGVs
Optimization	Without learning, only assigned behavior is performed. It means there is no optimal behavior selected	Accompanying with learning, UGV is able to select optimal behavior in each state. Therefore, optimizing task and status.

4. CONCLUSION

RACA works effectively with learning capability (based on RL) as tested in both numerical result and simulation. It also improves a performance of space robotic reconnaissance mission, contributing to an effective and autonomous collaboration between UGVs and UAVs, maximizing scientific data return and enabling fault tolerant mechanisms. Moreover, RACA is capable in extending for more tiers in M2S in the future. Therefore, it is assured that the architecture supporting Multi-tier Multi-agents scenario is proved to be excellently suitable for space exploration mission. Such a Multi-tier Multi-agent scenario provides solid performance in various aspects, such as reasonable time consumption, high reliability and optimization in cooperative strategy. It can be clearly stated that this achievement can be benefit to upcoming space reconnaissance mission, especially in term of increase of autonomy.

5. REFERENCES

[1]. B. Brummit and A. Stentz, GRAMMPS: A GENERALIZED MISSION PLANNER FOR MULTIPLE MOBILE ROBOTS IN UNSTRUCTURED ENVIRONMENT, Proceedings of the IEEE conference, Robots and Automation, Philadelphia, PA, 1988.

[2]. C. Boutilier, Planning Learning And Coordination In Multi-Agent Decision Processes,

Proceeding to 6th Conference Theory, Aspects of Rationality and Knowledge, pp. 195-210, Amsterdam, 1996.

[3]. C. Claus and C. Boutilier, The Dynamics Of Reinforcement Learning In Cooperative Multiagent Systems, Proceeding of AAAI-98, 1998.

[4]. C. Guestrin, M. Lagoudakis and R. Parr, Coordinated Reinforcement Learning, Proceeding to 19th International Conference on Machine Learning, pp. 227-234, 2002.

[5]. C. J. C. H. Watkins, Learning From Delayed Rewards. Ph.D. thesis, King's College, Cambridge, UK, 1989.

[6]. D. P. Bertsekas, Dynamic Programming: Deterministic And Stochastic Models, Prentice Hall, Englewood Cliffs, NJ, 1987.

[7]. D. Goldberg, V. Cicirello, M. Dias, R. Simmons, S. Smith, T. Smith and A. Stentz, A Distributed Layered Architecture For Mobile Robot Coordination: Application To Space Exploration, Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space, Houston, TX, 2002

[8]. G. Weib, Learning To Coordinate Actions In Multi-Agent Systems, Proceeding to IJCAI-93, pp. 311-316, Chambery, FR, 1993.

[9]. L. E. Parker, Alliance: An Architecture For Fault Tolerant Multi-Robot Cooperation, IEEE Trans. Robot. Automat, vol. 14, pp.220–240, 1998.

[10]. M. J. Mataric, Behavior-Based Control: Examples From Navigation, Learning, And Group Behavior, J. Exper. Theory. Artificial. Intelligence, vol. 9,no. 2–3, pp. 323–336, 1997.

[11]. M. Lauer and M. Riedmiller, Reinforcement Learning For Stochastic Cooperative Multi-Agent Systems, Proceeding of IEEE 3rd International Conference, Autonomous Agents and Multi-agent Systems, pp 1516-1517, 2004.

[12]. P.Pirjanian, T. Huntsberger, A. Barrett. Trplesent-Ing And Executing Plan Sequences For Distributed Multi-Agent Systems, Proceedings of IROS-2001, Maui, HI, 2001.

[13]. R. A. Brooks, A Robust Layerd Control System For A Mobile Robot, IEEE J. Robot. Automat, vol. 2, no. 1, pp. 14–23, 1986.

[14]. R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[15]. R. C. Arkin, Cooperation Without Communication: Multi-Agent Schema Based Robot Navigation, J. Robot. Syst., vol. 9, no. 3, pp. 351–364, 1992.

[16]. R. C. Arkin, Motor Schema Based Navigation For A Mobile Robot: An Approach To Programming By Behavior, Proceeding to IEEE International Conference, Robotics Automation, 1987, pp. 264–271.

[17]. S. Chien, R. Knight, A Stechert, R. Sherwood, and G. Rabideau, Using Iterative Repair To Improve The Responsiveness Of Planning And Scheduling, Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Breckenridge, CO, 2000.

[18]. S. Sen, M.Sekaran and J. Hale, Learning To Coordinate Without Sharing Information, Proceeding of AAAI-94, pp. 426-431, Seattle, 1994.

[19]. S.Singh, T. Jaakkola, M. L. Littman and C. Szepesvari, Convergence Results For Single-Step-On-Policy Reinforcement Learning Algorithms, Machine Learning, 1998

[20]. T. Estlin, D. Gaines, F. Fisher and R. Castano, Coordinating Multiple Rovers With Interdependent Science Objectives, Proceedings of ACM, AAMAS’05, Netherlands, 2005.

[21]. W. Fink and colleagues, Multi-Tier Multi-Agent Autonomous Robotic Planetary Surface/Subsurface Reconnaissance For Life, Lunar and Planetary Science XXXVII, 2006, 1433.pdf.