To appear in IF-3051 Strategy of Algorithm Technical Reports
Semester I - 2012

Multi-Robot Formation Control with Greedy and ϵ-Greedy Action Selection Reinforcement Learning Algorithm

Yudha Prawira Pane[#1]

13209073

[#] *School of Electrical Engineering and Informatics, Institut Teknologi Bandung*
Ganesa 10, Bandung 40132, Indonesia
[1]yudhapane@students.itb.ac.id

Abstract— **Multi-robot formation control has become an active research topic in the recent years. Several control system policy has been successfully implemented and tested. However, the AI approach to solve this problem is still rarely found. In this paper, author propose a reinforcement learning algorithm to control the formation of robots swarm. The method propose two action selection method that aims to maximize the agent's control policy and value function: greedy and ϵ-greedy.**

Keywords— **Multi-robot, formation control, reinforcement learning, greedy algoritm.**

I. INTRODUCTION

Formation control has become one of important aspects for multi-robot coordination. This method aims to alleviate the problems rising in deploying numbers of robot such as limited energy, time-inefficiency, unrobustness to uncertainty, among others. Concerning real-world application, formation control proposes a promising solutions for managing emerging issues like autonomous security patrols, search and rescue in hazardous environments, military applications, even tackling traffic congestion with car platooning.

The proclivity of today's control system research that digresses from single agent/system to multi-agent/system attracts researchers from around the world in designing an optimal control policy for such problem. The extensive efforts resulted in several solutions. Some of notable examples are as following: [1] proposes a fuzzy logic algorithm for enabling swarm of robots to avoid obstacles, [2] proposes a optimized, distributed control algorithm, meanwhile [3] uses a behavior based control architecture in order to achieve a desired formation.

However, there is only a few researches that implement machine learning algorithm for solving formation control problem. In fact, machine learning has an advantage in managing uncertainty about the robots' and environment's model furthermore providing a method to improve the robots' performance over time. This motivates a further investigation for integrating learning, particularly reinforcement learning, into multi robot formation control.

This paper propose a solution that integrates reinforcement learning algoritm for controlling a swarm of mobile robots. In order to achieve an optimal action-value equation, the reinforcement learning is optimized with greedy and ϵ-greedy algorithms for action selection. This paper designs the method and introduces comparison between both algorithms.

The rest of the paper is organized as follows. In section II, the kinematics model consisting position, orientation, and velocity of mobile robot is derived. Section III explains the control problem that becomes the main issue to address. Section IV focuses on the idea of greedy algorithm. In section V, reinforcement learning is introduced and explained in detail. Section VI develops the greedy and ϵ-greedy algorithm for formation control. Finally, the paper is concluded in section VII.

II. KINEMATICS OF MOBILE ROBOT(S) AND THE CONTROL SYSTEM DESIGNS

A. Kinematic Model of Mobile Robot

Kinematic model consists of the position, orientation and each one's derivative (linear and angular) of mobile robot over time. Position means its (X,Y) coordinate relative to the world reference frame. Orientation means its angle with respect to X^+ axis of world reference frame. To get more insightful comprehension, these properties are depicted in figure 1 below.

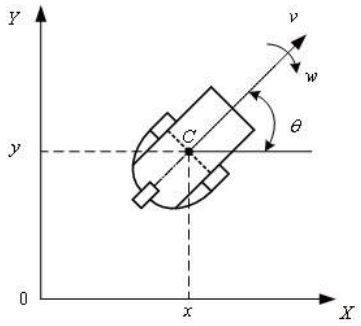

Figure 1. Kinematic properties of mobile robot. The position is represented by (x,y) value or center of robot's coordinate. The orientation is represented by angle Θ.

In eucleidan space (X-Y coordinate system), the position and orientation of the robot is a function of previous position and the respected velocity. This, also called the robot's state, is mathematically denoted as follows.

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} v(k) + v(k)Tcos\theta(k+1) \\ y(k) + v(k)Tsin\theta(k+1) \\ \theta(k) + \omega(k)T \end{bmatrix}$$

Where $(x(k), y(k))$ is the robot's position, $\theta(k)$ is the orientation, $v(k), \omega(k)$ is the robot's linear and angular velocity respectively, and T is sampling time of the state signal. In multi-robot formation, we want to control the robots' position and orientation to construct a particular formation (eg. triangular, trapezoidal , etc).

B. Formation Control via Leader-Follower Approach

One of the most common approached used for controlling multi-robot formation is leader-follower schema. This approach assumes that only local sensor-based information is available for each robot. In defining the controller feedbacks, there are two types of configuration: $l - \psi$ and $l - l$ controller. Both are shown in figure 2 and 3 below.

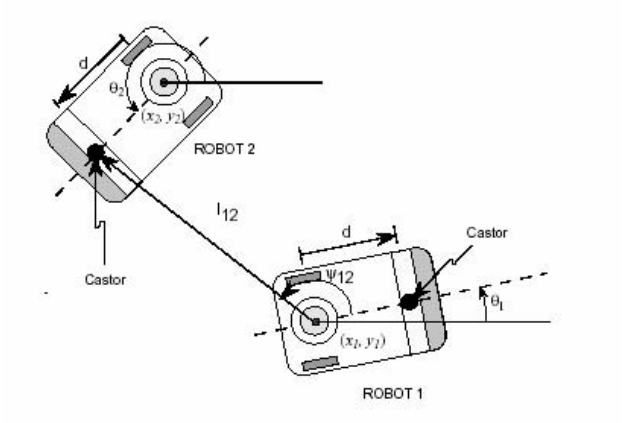

Figure 2. The $l - \psi$ controller configuration

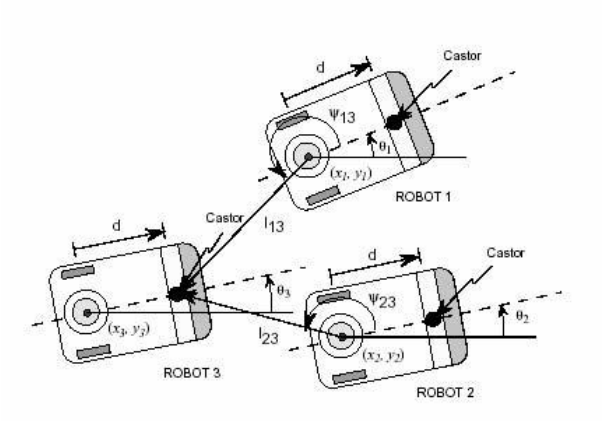

Figure 3. The $l - l$ controller configuration

The $l - \psi$ controller maintain the l and ψ value of two mobile robots (see fig.2) in order the keep the formation as required. Meanwhile the $l - l$ controller maintains the position of three mobile robots from the value of l_{13}^d and l_{23}^d (see fig. 3). In this paper, the $l - \psi$ type controller is used.

III. PROBLEM DESCRIPTION

The problem is defined as follows. Given multiple robots, a desired formation and an environment with obstacles, how should the robots move through obstacles while maintaining its formation with smallest error possible. To make the case more concrete, the problem chosen is maintaining a triangular formation with a customize controller configuration. The obstacle in the environment takes a random shape, approximating an oval or circular shape.

To form a triangular formation, three robots is deployed where each one has an information about the relative position of two other robots. This paper takes into account the communication network limitation as well, where two robots can only send data to each other within a defined range of length, say d1 – d2 meters. If the two robots are separated more than d2 meters, then the communication is lost/broken. In the other hand, if the robots are too close (i.e. less than d1 meter) then the signals face interference problem resulting in lost communication as well.

The triangular formation is an equilateral triangle with each side length equals d. The three robots have to move together, forming the triangle, through obstacles. Scenario of this particular task is given in figure 4.

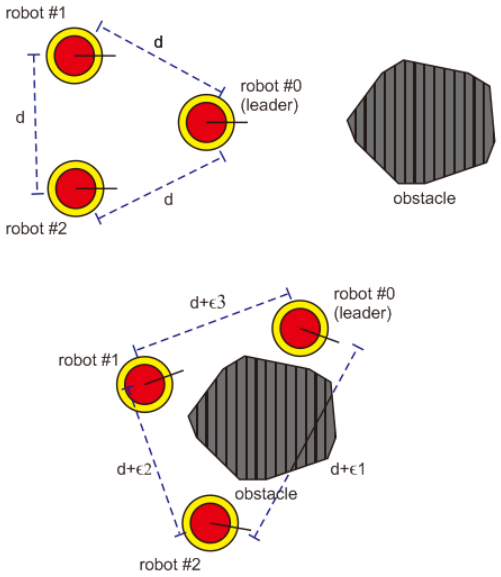

Figure 4. Triangular formation control obstacle scenario. Initially, the robots form an equilateral triangle with side d. When moving through obstacle, triangle is still maintained but with side errors.

IV. Greedy Algorithm

Greedy algorithm is an algorithm that is designed to solve optimization problem. This algorithm is simple and straightforward. The greedy algorithm maximizes immediate value/reward without regarding future events by forming step by step solution. Thus, greedy algorithm only aim to achieve local optimum.

Despite its simplicity and rather-trivial characteristic, this algorithm has proven to be a quite successful tool for solving some optimization problems, specifically when there is a contraint for computational resource. To implement greedy algorithm, there are several properties that must be defined:

1. Candidate Set

This consists of elements that forms the solution. For instance, job set, vertices in a graph set, etc.

2. Solution Set

This consists of candidates chosen as the solutions. In other words, solution set is a subset if candidate set.

3. Selection Function

A function for selecting the most possible candidates for achieving optimum solution.

4. Feasibility Function

A function for checking if a chosen candidate will result in a feasible solution i.e. not violating the contraints.

5. Objective Function

A function for maximize or minimize the solution value e.g. path length, income, etc.

The pseudo-code for greedy algorithm is as follows:

```
function greedy ( input C: candidate_space) → candidate_space

Declaration:
    x: candidate
    S: candidate_set

Algorithm:
    S ← {}
    while (not SOLUTION(S)) and (C≠{}) do
    X ← SELECTION(C)
    C ← C – {x}
      if FEASIBLE(S U {x}) then
          S ← S U {x}
      endif
    endwhile

    {SOLUTION(S) or C= {}}

    if SOLUTION(S) then
          return S
    else
          write (“no solution exists”)
    endif
```

V. Reinforcement Learning

A. *Reinforcement Learning Fundamental*

Reinforcement learning (RL) is a type of machine learning algorithm that can learn the optimum policy for an agent without training samples provided. The algorithm is learning by interacting with the environment. An RL agent learns from the consequences of its action. Rather than being taught explicitly, an RL agent selects its action on basis of its past experience and also by new choice resulted from exploration. One of the way to learn is by performing trial and error learning.

Reinforcement learning consists three main subelements/ properties which explained below.

1. Policy

A policy maps each state of an RL agent (in this case multiple robots), s € S, and action, a € A(s), to the probability π(s,a) of taking action a in state s. The agent use the policy in making decision based on what it perceives from the environment. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of reinforcement learning agent in the sense that it alone is sufficient to determine the agent's (robot) behavior.

2. Reward Function

The goal of reinforcement learning algorithm is defined as a reward function, R(t). At each time step, the agent will receive a reward value based on its achievement. This reward could be positive (the robot's performance has improved) or negative (the robot performs worse, hence punishment). The rewards which is formed over a long period is called value.

3. Value Function

Value function is a result of reward function that is applied for a long run. It specifies which state/action is *good* or *bad*. Whereas rewards determine the immediate, desired environment/robot's state, value represents the *long-term* desirability of the robot's state.

In order to form an optimal and accurate value function, the robot needs to know an accurate representation of its state as well. When the representation of the environment's state is accurate or complete, the state satisfies Markov property. Such reinforcement learning task is also called Markov Decision Process (MDP). In determining the value function, an RL agent's MDP must be defined first.

Assuming that the robot's MDP is already given, the value function equation can be derived as follows.

$$V^{\pi}(s) = E_{\pi}\{R_t|s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\}$$

Where $V^{\pi}(s)$ is the expected return when startig in state s and following policy π and E_{π} denotes the expected value given that the agent follows the policy. We call the function V^{π} the *state-value function for policy* π.

Finally, we come to the value function which is mathematically defined below.

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\}$$
$$= E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\}$$

We call $Q^{\pi}(s,a)$, value of taking action a in state s under policy π, *the action-value function for policy* π. Value function satisfy particular recursive algorithm:

$$V^{\pi}(s) = E_{\pi}\{R_t|s_t = s\}$$
$$= E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\}$$
$$= E_{\pi}\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\}$$
$$= \sum_{a}\pi(s,a)\sum_{s'}P^a_{ss'}\left[R^a_{ss'}\gamma E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_{t+1} = s'\right\}\right]$$
$$= \sum_{a}\pi(s,a)\sum_{s'}P^a_{ss'}\left[R^a_{ss'} + \gamma V^{\pi}(s')\right]$$

B. Action Value Method

It has been stated before that long time rewards form value function. Intuitively, to obtain a value function that maximizes robot's performance, the total reward must also be maximized. However, maximum reward in a given state transition or action performed does not guarantee that the final value function will be maximum. In the other hand, a low reward value does not necessarily mean that the final value function will be eventually low as well. This raise a question concerning how to select action (in this case, how to move the robots) in order to maximize the value function. How should we select reward in each step in order to get an optimum accumulation in the end. It turns out that there are already several techniques to answer this action selection problem. Popular ones are greedy, ϵ-greedy, softmax action selection, evaluation vs instruction, incremental, optimistic initial value, pursuit methods, etc. Greedy and ϵ-greedy algorithm become focuses in this paper.

VI. Robots' Formation State Representation and Algorithm Design

First, we define the state of the robots' formation. Because we use the $l - \psi$ controller configuration, there are total

6 state variables: $l_1, l_2, l_3, \psi_1, \psi_2, \psi_3$ (see fig 5). We can make the representation more compact by using a 6x1 matrix below:

$$\text{State } s(t) = \begin{bmatrix} l1(t) \\ l2(t) \\ l3(t) \\ \psi1(t) \\ \psi2(t) \\ \psi3(t) \end{bmatrix}$$

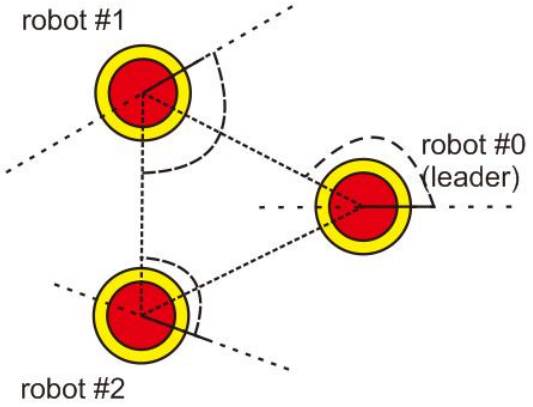

Figure 5. All elements that form robots's state. There are total 3 distances (forming a triangle) and 3 angle values.

Given the triangular properties (d, dmax, dmin and 60^o angle) , we can set the reward for the robot as a function of its formation error. The error is defined for several cases.

1. The distance of a pair of robots exceeds d, but not exceeds dmax
2. The distance of a pair of robots is lower than d, but not lower than dmin
3. The distance of a pair of robots exceeds dmax
4. The distance of a pair of robots is less than dmin
5. The angle formed by a pair of robot does not equal 60^o.

Those error cases results in negative reward (punishment). Meanwhile, the robots receive positive rewards when the difference between its errors (error in current state minus error in previous state) is positive.

if error(t+1) – error (t) < 0
then r(t) > 0 {robot perform better}

else if error(t+1) – error (t) > 0
then r(t) < 0 { robot perform worse}

else r(t) = 0

Action selection algorithm is responsible in selecting actions that would maximize rewards, and hopefully maximize value as well. The action for any given state can consists of infinite numbers (since the movement control is in form of parameterized function). But in more coarse representation, the actions could fall in several categories:

1. Move forward
2. Move diagonally
3. Move backward
4. Turn/Pivot left
5. Turn/Pivot right

A. *Greedy Action Selection Algorithm*

In greedy action selection, the algorithm select the maximum reward for each state and chooses the corresponding action. This will result in local maximum for the reward value but does not guarantee an optimum value function.

B. *ϵ- greedy Action Selection Algorithm*

In ϵ-greedy action selection algorithm, the robot will sometimes pick randomly the reward that is not maximum. This enables the robot to conduct exploration in searching for the best policy. ϵ is represented as a value ranges from 0 to 1. The highest the value, the closer it approach a standard greedy method. The expected results for optimum value function is depicted in figure 6 below.

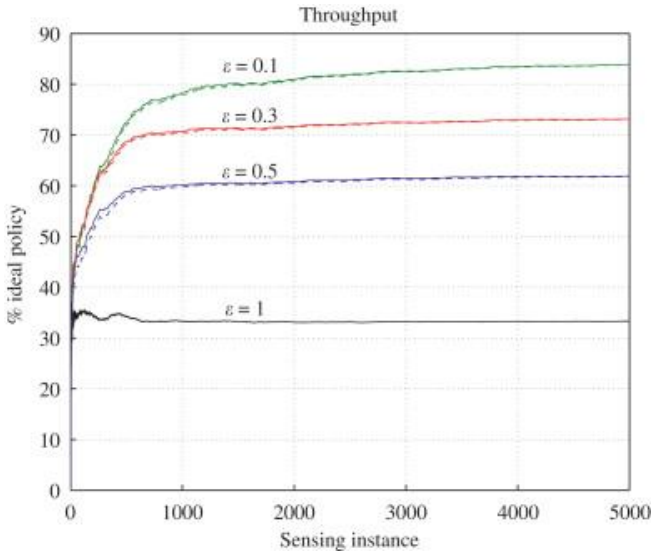

Figure 6. The comparison of greedy method and ϵ-greedy method. The greedy method (ϵ = 1) resulted in a stagnant value function. Meanwhile the ϵ-greedy method have better performance

VII. CONCLUSION

1. Reinforcement learning provides a promising solution to solve multi-robot formation control problem

2. In selecting the action to maximize the value function, an RL agent can use two methods among others: greedy and ϵ-greedy.
3. ϵ-Greedy has proven to be better than greedy for various reinforcement learning application, but extensive experiment for multi robot formation control is needed.

REFERENCES

[1] Bazoula, A. Djouadi, M.S., Maaref, H., *Formation Control of Multi-Robots via Fuzzy Logic Technique*. 2008 International Journal of Computers , Communication & Control

[2] Sutton, R.S, Barto, A.G. *Reinforcement Learning: an Introduction*, 2nd ed.MIT Press.

[3] Ren, W., Sorensen, N. *Distributed coordination architecture for multi-robot formation control.* Transaction of Robotics and Autonomous Systems

[4] Balch, T., Arkin, R. *Behavior based Formation Control for Multi Robot Teams*. IEEE Transcation on Robotics and Automation

[5] Munir, R. *Diktat Kuliah Strategi Algoritma*. Prodi Teknik Informatika, STEI ITB.