*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

# Reinforcement Programming: A Function Based Reinforcement Learning

Irene Erlyn Wina Rachmawan, Ali Ridho Barakbah, Ira Prasetyaningrum, Yuliana Setiowati
Information And Computer Engineering, Electronic Engineering Polytechnic Institute Of Surabaya
ireneerlyn@student.eepis-its.edu,ridho@eepis-its.edu,ira@eepis-its.edu,yuliana@eepis-its.edu

**Abstract**

*Reinforcement Learning (RL) is a machine learning technique that solves problem by using exploration and exploitation method based on specific goals that has been initiated. This paper present a new approach that allows the Reinforcement Learning (RL) solves case base optimization, by using the RL behavior in considering exploitation and exploration. This new approach, called Reinforcement Programming (RP), builds by modifying RL, in which RL methods are modified by adding variable alpha, beta as new formula for learning method. Alpha and beta is subset of RP in finding solution of case base optimization and with specific value of alpha and beta it will impact the accuracy of solution. In this paper we implement and give a testing to our proposed algorithm for optimizing K-means centroid optimization. The Result of our proposed idea has been compared into several optimization algorithm. The experimental results show the improved solution of case based function using proposed approach.*

Keywords: Reinforcement Learning, Reinforcement, Optimization, Algorithm.

## 1. Introduction

The recent learning theory that has been developed by many scientific is theory of learning based on supervised learning. Supervised learning is the machine learning task of inferring a function from labeled training data. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. Thus, did not create a true-learning machine which can improve the ability of agent to solve problems that given to it, but create a learned machine, a smart machine after had some training method.

Reinforcement Learning is a new paradigm in learning theory, which applied into a machine or a computer system to make a smart machine or computer. Inspired by human behaviorist psychology, reinforcement learning concerned with how software agents ought to take actions in an environment and collect as many reward as possible to reach its goal [1]. The agent decides the action that will be taken by considering exploitation and exploration. Exploitation is a process to get as many information as possible from its owned library or experienced. Exploration is a process for collecting new information from its environment by exploring a state that has been de-fined in environment [2].

Reinforcement learning combines supervised and unsupervised learning theory that makes Reinforcement Learning interesting to learn because of its behavior for become more intelligent after learning from its environment. Reinforcement Learning improves its ability after collecting reward for each step for solving a problem. Thus, make Reinforcement Learning as true-machine learning which can be smarter after learning by itself.

The case based optimization literature is also vast [3]. This case problem and its variants are used in many resource management applications such as cargo loading, industrial production, menu planning, and resource allocation in multimedia servers. In this paper we will specify case in this paper and discussing finding prince as problem testing. The finding cases prince is a variant of heuristic problem. Here we have to find prince position by using distance formula between agent and the prince as fitness value. In iteration process the agent will take and action to shortening distance to princess. A good solution is decided with the final distances between agent and prince.

Until recently, exact methods for the solving case based optimization were dominated by methods by generate solution though random number value in several state such as Genetics algorithm and ant Colony. The random number will cause different solution in different experiment.

*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

Reinforcement Learning helps in improving the performance of agent to solve a case. Reinforcement Learning is a relatively new approach to problem solving that takes inspiration from the behavior of human brain. In particular, human brain will learn by exploit a new condition and save the result as reward [4]. So human brain will also have an experience based on exploitation. Therefore (RL) is also a very successful Artificial Intelligence subarea. RL algorithms are very useful for solving a wide variety problems when their models are not available a priori, since many of them are known to have guarantees of convergence to equilibrium. However, the convergences of a RL algorithm may only be achieved in solve case based goal not case based function. Therefore in this paper we proposed new algorithm to implement Reinforcement Learning cycle in case based function.

## 2. Basic Theory of Reinforcement Learning

Reinforcement learning is a way of getting an agent to learn [5]. The agent learns by receiving rewards after every action. It somehow keeps track of these rewards, and then selects actions that it believes will maximize the reward it gains, not necessarily only for the next action, but in the long run. The agent usually goes through the same environment many times in order to learn how to find the optimal actions. Balancing exploration and exploitation is particularly. Because the agent may have found a good goal on one path, but there may be an even better one on another path [6]. Without exploration, the agent will always return to first goal, and the better goal will not be found. Or, the goal may lie behind very low reward areas, which the agent would avoid without exploration. On the other hand, if the agent explores too much, it cannot stick to a path; in fact, it is not really learning: it cannot exploit its knowledge, and so acts as though it knows nothing. Thus, it is important to find a good balance between the two, to ensure that the agent is really learning to take the optimal actions [7].

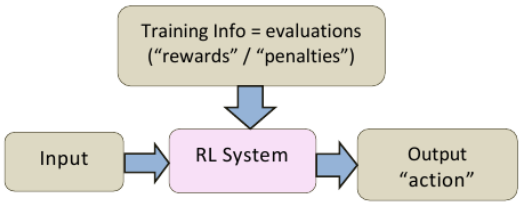


**Figure 1.** The basic reinforcement learning scenario.

The basic scenario of Reinforcement Learning has objective to get as much reward as possible [8]. Reward or punishment will declare in RL system. Which every optimization case has different rule to decide the rewards and penalties for action of Reinforcement Learning agent. The output is an action that will take by agent to solve problem by considers the whole problem of a goal-directed.

Reinforcement learning by interacting with an uncertain environment and decide what to do-how to map situations to actions-so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.The policy is some function that tells the agent which actions should be chosen, and it is learned through trial-and-error interactions of the agent with its environment [9].

## 3. Our Proposed Reinforcement Programming

Reinforcement Programming (RP) algorithm is the algorithm using basic concept of reinforcement learning. So in its implement, Reinforcement programming has the same behavior of Reinforcement Learning, by involving a balance between exploration of uncharted territory and exploitation of current knowledge to find solution. The solution determined by as much reward as possible in process learning. Reward and punishment are a value given by environment from the agent step.

In RP, this policy is learned through trial-and-error interactions of the agent with its environment: on each interaction step the agent senses the current states of the environment, chooses an action to perform, executes this action, altering the states of the environment, and receives a scalar reinforcement signal r (a reward or penalty).

The benefits of Reinforcement Programming are to bring a benefit in optimization case using an intelligent learning approach based on Reinforcement Learning. With involving the characteristics of Reinforcement Learning, Reinforcement Programming provides an experience-based learning to achieve the global optimum.

### 3.1. Variable Reinforcement Programming

Reinforcement Programming is mainly based on the Reinforcement Learning. A number of slight modifications of Reinforcement Programming can be formulated where:

- β : is a variable value to give impact to step that the agent will take.
- μ : a variable that will set a step value to the last step.

The goal of the agent in a RP problem is to learn an optimal solution by set $action \leftarrow d_p \,.\, step_p$ that action is accumulated from previous step with direction of step that will be taken. The current States will be assign in variable $newS \leftarrow$.New states will be accumulated by

*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

cuttent new states and action $newS_p \leftarrow newS_p + action$ ($newS_p$ must be in $[minval_p..maxval_p]$). And assign variable step with rule :

$$newS_p \leftarrow S_p + action \quad (1)$$

The RL technique is well-known uses a strategy to learn an optimal via learning of the action values. It iteratively approximates new States. In RP the condition of using exploitation or exploration are decided by random

$$step_p \leftarrow step_p - \mu \, . \, r_p \, . step_p \quad (2)$$

p are probability to take action whether to exploit or explore a finite state. To balance exploitation and exploration p can be set in 0.5. Variable on reinforcement Programming structure is given in figure 2.

| | | | | | |
|---|---|---|---|---|---|
| State | $S_1$ | $S_2$ | $S_3$ | … | $S_n$ |
| Reward | $r_1$ | $r_2$ | $r_3$ | … | $r_n$ |
| Direction | $d_1$ | $d_2$ | $d_3$ | … | $d_n$ |
| Step | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | … | $\alpha_n$ |
| minval | $min_1$ | $min_2$ | $min_3$ | … | $min_n$ |
| maxval | $max_1$ | $max_2$ | $max_3$ | … | $max_n$ |

**Figure 2.**The basic variable reinforcement programming.

The main RP variable that has an important condition to solve optimization problem can be explain below:

- State: is a finite set of states.
- Reward: a variable to save reward and punishment.
- Direction: is a direction that will take agent to the optimum solution.
- MinVal, MaxVal : is a finite area of cases that the agent can perform.
- Action Value (av) : variable that save the value of agent action.

State is the first position of agent to start solving problem. State can be initiate as assigned value or random number, depends of cases that will handled by agent. Variable reward is an array to save reward of each state that has been declared. To change the action direction of agent we need to declare direction variable. In first position direction can be assign with positive number to increase the direction in positive grid line. Action value is an array to save the calculation of action that has been taken by agent.

Variable above are formulating to have compatible with optimization cases. Modification of different heuristic case will change the condition of variable state and direction.

3.2. System Architecture

System architecture given in figure 3, explain the flow of Reinforcement Programming while find the best solution from optimization problem. The basic Reinforcement Programming algorithm starts with an initialization phase, where

i. Assign data item and set into variable dataset
ii. Set modeling state value calculation(depend on optimization cases)
iii. Set probability for exploration rate. Use 0.5 to get a balanced action for exploration and exploitation.
iv. Assign the value of new state in state.
v. The agents process state evaluation to whether receive reward or punishment for current action.

This is done using an index that stores the positions of all 'free' data items on the grid. The process is as follows:

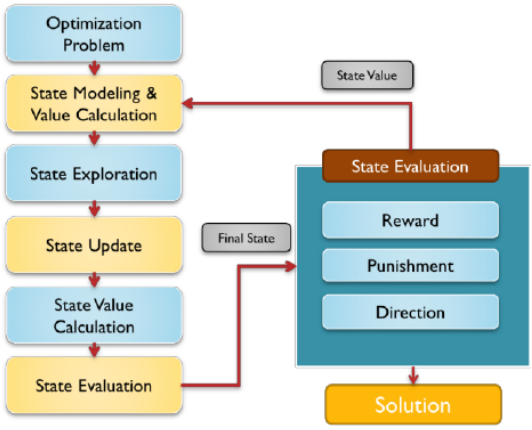


**Figure 3**. Step of reinforcement programming.

First step, RP processing case based function by modeling state and state value calculation. Then agent will begin state exploration in finite environment of problem, the actions that can be taken by agent is determined by exploration rate. If a random number is bigger than exploration rate than the agent will choose solution randomly in finite area. But if the random numbers are smaller than exploration rate than agent will consider taking an action based on reward. After take an action agent updates the current state and calculates current state value. A value from current state will label as reward or punishment, and determining the next

*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

direction of exploration. If agent receives punishment, then the agent will change its direction for the next exploration step. This process will be iterates in some value that already assign as number of learning time for agent of RP.

An extension of this algorithm is algorithm is presented where the parameter is an adaptively updated during the execution of the algorithm. This algorithm is given in Figure 3.

```
(1)  Procedure Reinforcement Programming
(2)    Initialize dataset on the toroid grid
(3)    Randomly place agent on toroid grid
(4)      For I = 1 to max_iteration
(5)      Calculate sv
(6)      R = random exploration number
(7)      If (r > exploration_rate)
(8)        Let agent randomly explore environment
(9)      Else
(10)       Let agent use its knowledge to determine the next
           action by the biggest reward value
(11)     End If
(12)     Execute action
(13)     Calculate new state value
(14)     Calculate new step position // see equitation (2)
(15)     If(newsv > sv)
(16)       Update array S
(17)       Increase and Update reward value // see
           equitation (3)
(18)     Else if (newsv < sv )
(19)       Set punishment and update reward value //see
           equitation (4)
(20)       Change direction
(21)     Else
(22)       Let agent change direction to explore environment
(23)     End If
(24)   End for
(25)  End Procedure
```

**Figure 3**. Reinforcement programming algorithm.

Reinforcement Learning needs to initialize dataset to identify problem environment. Agent will place randomly or assigned value depends on handled cases. After calculate state value, agent will determine action between exploration and exploitation. Then execute action and calculating new state value to determine reward based on state current value. Reward of new state in the environment is computed through the following Equation (3) and (4)

$$r_p \leftarrow r_p + \beta \,.\, (1\text{-}r_p) \qquad (3)$$

The equation above show the formula for increasing reward value.

$$r_p \leftarrow r_p - \beta \,.\, (1\text{-}r_p) \qquad (4)$$

And in case to give punishment, Reinforcement Programming used formula above to decrease reward value. Where $\beta$ is a constant to set a value of reward with scalar 0.1 until 1. The more scalar that will be used it will impact the increase or decrease value of reward.

The output of Reinforcement Programming algorithm is a solution of given problem.

## 4. Experiments in the Optimization Cases

Reinforcement Programming experiment has been tried on Finding Princes as initial optimization case. Finding Princes are most simple optimization techniques. Starting from a random position of agent, the algorithm repeatedly computes the distance to reach a better and closest point to prince or goal position.

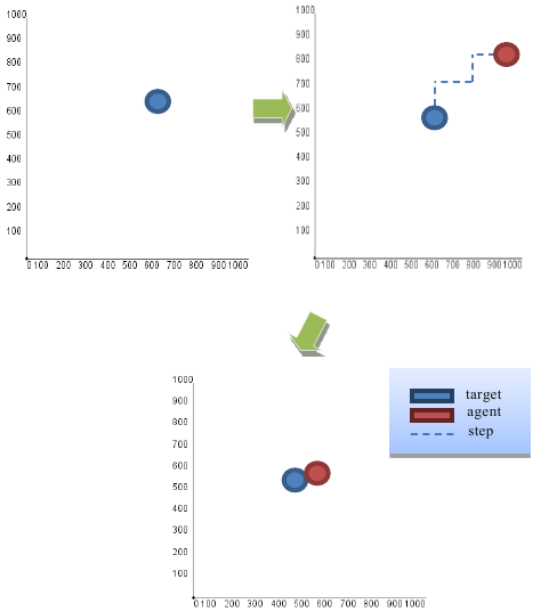


**Figure 4**. Finding prince basic algorithm.

The algorithm for the simple Finding Princes is given as follows:

1. Set position of princes
2. Assign random position of agent.
3. When all objects have been assigned, recalculate the positions of the agent to princes' position.
4. Go back to Steps 2 until iteration limit

The initial agent is normally set in randomly position as Finding Princes standard algorithm. Agent will continuously search the closest position with closest distances to prince's position. We implement our proposed idea to solve Finding Princes algorithm. With the first condition in experiment is agent being

*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

positioning in random finite area, then in N iteration agent need to take action to get as close as possible to prince position. Agent will move into a new state to solve Finding Prince problem, the current position of agent will calculated with distance formula to get the fitness value of current agent.

When this assignment process is over, a new position will be calculated for distances with prince's position. The closest current agent position in experiment result will be noted and analyzed in table of experiment.

This paper shows about Reinforcement Programming compare with Genetic Algorithm[10].

4.1. Data Plot Example

The data plot of Finding Prince are given with variable bellow

**Table 1**. Variable of finding prince.

| Prince Position | | Finite Area | |
|---|---|---|---|
| X | Y | X | Y |
| 50 | 50 | 1000 | 1000 |

The experiments are performed over a mathematically generated 2D data plot. Figure 4 shows the data plot distribution. Dataset 1 is made based on a mathematical model to form princes position in finite area

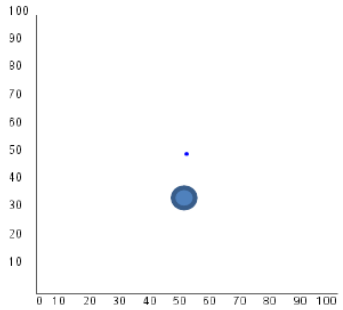


**Figure 4**. Finding prince data.

Data plot Finding prince consist one point scattered around finite area. This data plot will be used as an evaluation value of current state of agent, which has a function to find the shortest distance with prince position. The data plots above are to be tested using Reinforcement Programming and Genetic Algorithm. The average error and average time taken by the Finding Prince with GA approach is quite high. Finally, running Reinforcement Programming to solve Finding Prince definitely guides to the best solution among the group, this appears clearly from the results obtained. The following Figure 6 and 7 will show the last position from running algorithm. Each algorithm use number 100, 200, 500, and 1000 as max iteration.

The experiment using 100, 200, 500 and 1000 times of iteration, in each algorithm runs in 5 times of experiment. The accuracy of current state will count by the distance with prince position and two variables: Error ratio and time consuming. In experiment noted the number of valuing and time consuming for each algorithm.

GA algorithm used 10 individual in one population and the operator of cross over and mutations are direct switch.

**Table 2**. Genetics Algorithm Result

| iteration | Final X | Final Y | Error Rate | t |
|---|---|---|---|---|
| 100 | 60.9967406 | 65.7679974 | 26.76474 | 21 |
| 100 | 41.5547915 | 74.3364451 | 32.78165 | 19 |
| 100 | 23.9424198 | 79.9420789 | 55.99966 | 20 |
| 100 | 55.0478627 | 28.0488782 | 26.99898 | 18 |
| 100 | 44.9324645 | 57.5445519 | 12.61209 | 19 |
| Sum | 226.474279 | 305.639951 | 155.1571 | 97 |
| Average | 45.2948558 | 61.1279903 | 31.03142 | 19.4 |
| 200 | 50.3555437 | 77.6294531 | 27.985 | 35 |
| 200 | 38.0749452 | 48.3612969 | 13.56376 | 37 |
| 200 | 35.1263177 | 49.9391183 | 14.93456 | 36 |
| 200 | 59.633487 | 36.5777629 | 23.05572 | 36 |
| 200 | 43.3017596 | 55.8329198 | 12.53116 | 36 |
| Sum | 226.492053 | 268.340551 | 92.0702 | 180 |
| Average | 45.2984107 | 53.6681102 | 18.41404 | 36 |
| 500 | 54.6021979 | 38.8958947 | 15.7063 | 74 |
| 500 | 40.085668 | 40.085668 | 19.82866 | 74 |
| 500 | 30.4706487 | 61.6649164 | 31.19427 | 68 |
| 500 | 55.8238653 | 48.1592435 | 7.664622 | 71 |
| 500 | 54.6117249 | 40.0515771 | 14.56015 | 62 |
| Sum | 235.594105 | 228.8573 | 88.954 | 349 |
| Average | 47.1188209 | 45.77146 | 17.7908 | 69.8 |
| 1000 | 53.1414055 | 74.9771694 | 28.11857 | 97 |
| 1000 | 51.9714817 | 66.69961 | 18.67109 | 98 |
| 1000 | 61.2387989 | 55.2687491 | 16.50755 | 98 |
| 1000 | 59.9354446 | 49.6764192 | 10.25903 | 99 |
| 1000 | 56.5181311 | 56.7387422 | 13.25687 | 97 |
| Sum | 282.805262 | 303.36069 | 86.81311 | 489 |
| Average | 56.5610524 | 60.672138 | 17.36262 | 97.8 |

*The Third Indonesian-Japanese Conference on*
*Knowledge Creation and Intelligent Computing (KCIC) 2014* *ISBN : 978-602-9494-97-6*

As for Reinforcement Learning the value of variable that used in experiments are: miu: 0.1, beta: 0.01, exploration_rate = 0.3;

**Table 3**. Reinforcement Programming Result

| iteration | Final X | Final Y | Error Rate | t |
|---|---|---|---|---|
| 100 | 49.1061963 | 50.6793765 | 1.573180225 | 2 |
| 100 | 47.0370931 | 56.5863569 | 9.549263728 | 2 |
| 100 | 48.6825575 | 50.3564682 | 1.673910722 | 2 |
| 100 | 43.4562944 | 48.9193939 | 7.624311662 | 2 |
| 100 | 51.9963295 | 51.9963295 | 3.992659035 | 2 |
| Sum | 240.278471 | 258.537925 | 24.41332537 | 10 |
| Average | 48.0556942 | 51.707585 | 4.882665074 | 2 |
| 200 | 49.7146325 | 45.6156134 | 4.669754109 | 4 |
| 200 | 50.1958153 | 48.6605951 | 1.535220279 | 4 |
| 200 | 48.6825575 | 50.3564682 | 1.673910722 | 4 |
| 200 | 50.3254017 | 45.5039659 | 4.821435832 | 4 |
| 200 | 51.7855587 | 49.6374001 | 2.148158674 | 4 |
| Sum | 250.703966 | 239.774043 | 14.84847961 | 20 |
| Average | 50.1407931 | 47.9548085 | 2.969695923 | 4 |
| 500 | 50.1205468 | 49.0707099 | 1.049836886 | 7 |
| 500 | 49.9806405 | 50.0868376 | 0.106197084 | 7 |
| 500 | 50.0392062 | 50.0409866 | 0.080192867 | 7 |
| 500 | 50.0956309 | 50.1240767 | 0.219707624 | 7 |
| 500 | 50.0694342 | 50.0112215 | 0.080655739 | 7 |
| Sum | 250.305459 | 249.333832 | 1.5365902 | 35 |
| Average | 50.0610917 | 49.8667665 | 0.30731804 | 7 |
| 1000 | 50.0246154 | 50.8772517 | 0.901867081 | 17 |
| 1000 | 50.0877358 | 49.9223708 | 0.165365021 | 18 |
| 1000 | 50.0706273 | 50.119774 | 0.190401326 | 18 |
| 1000 | 50.0342252 | 50.0362856 | 0.070510831 | 19 |
| 1000 | 50.1545055 | 50.0151317 | 0.169637255 | 17 |
| Sum | 250.371709 | 250.970814 | 1.497781513 | 89 |
| Average | 50.0743418 | 50.1941628 | 0.299556303 | 17.8 |

Table 2 shows the performance of K using Genetics algorithm, and shows average error are quite high.From table 3 shows it is clear that the proposed approach shows very less error rate compared to other methods.

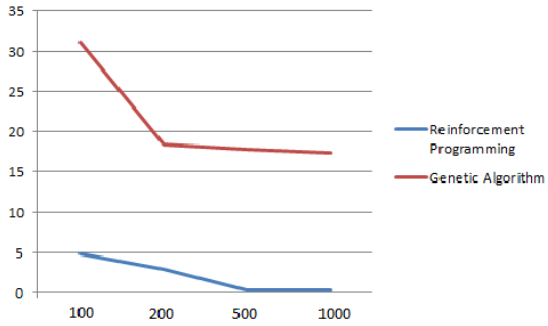


**Figure 5**. Finding Prince Data

As we can see from graph above shown that Reinforcement Learning faster in find solution and the number of iteration also impact the number of accuracy in finding solution.

## 5. Contribution and Conclusion

Reinforcement Learning is powerful algorithm inspired by human brain in solving problem by using exploration and exploitation. In other hand Reinforcement Learning only can implement on case based goal.

To use RL benefit to solve case based optimization, this paper proposed Reinforcement Programming as new algorithm. The intention of using RP is its capability of convergence and it follows the behavior of RL. The experimental evaluation scheme was used to provide a common base of performance assessment and comparison with other methods. Finally, when comparing the experimental results of Finding Prince optimization initiate centroid by using Genetic Algorithm and Reinforcement Programming it is observed clearly that optimizing Finding Princes with Reinforcement Programming is better than the simple genetic algorithm. As shown by the results on all data plot finding prince with Reinforcement Learning is ready to achieve high clustering accuracy if compared to other algorithms. Time consuming in each trial also varies depends on alpha value, which used for exploration rate.

Although function cased program can be solved by several function based algorithm such as Genetic Algorithm. The precision value are can be different in each evolutionary. So the use of Reinforcement Programming can produce a stable solution, and high precision. This lead people to get an optimum solution of Function-based cases.

**References**

[1] Bram Bakker& Jurgen Schmidhuber, HierarchiCal Reinforcement Learning Based On Subgoal Discovery And Subpolicy Specialization. *Technical report, IDSIA, 2003*

[2] Ming tan, Multi-Agent Reinforcement Learning : independentvs cooperative agent, *GTE Laboratories Incorporated*

[3] C.Immaculate Mary, dr. S.V. Kasmir Raja, REFINEMENT OF CLUSTERS FROM K-MEANS WITH ANT COLONY OPTIMIZATION, *Journal of Theoretical and Applied Information Technology, 2005 – 2009*

[4] Peter Dayan, Christopher JCH Watkins. Reinforcement Learning, *Encyclopedia of Cognitive Science*

[5] Abhijit Gosavi, A Reinforcement Learning Algorithm Based On Policy Iteration For Average Reward: Empirical Results With Yield Management And Convergence Analysis, *Kluwer Academic Publishers,2004*

[6] Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore. Reinforcement Learning: A Survey, *Journal of Articial Intelligence Research 4*

[7] Reinaldo A. C. Bianchi, Raquel Ros and Ramon Lopez de Mantaras, Using Case Based Heuristics to Speed up Reinforcement Learning.

[8] Csaba Szepesvari, Algorithms for Reinforcement Learning, *Morgan & Claypool Publishers,2009*

[9] Paul Wagner, Richard S. Sutton and Andrew G. Barto, *Journal of Theoretical and Applied Information Technology, 2006*

[10] Michael D. Vose, The Simple Genetic Algorithm, *Massachussets Institute of Technology, 1999*