computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

A MULTI-AGENT APPROACH TO POMDPS USING OFF-POLICY REINFORCEMENT LEARNING AND GENETIC ALGORITHMS

Samuel O. Obadan, Zenghui Wang

College of Science, Engineering and Technology, University of South Africa, Florida 1710, South Africa
obadansam@gmail.com, wangzengh@gmail.com

Paper history:
Received 22 November 2019
Received in revised form 19 March 2020
Accepted 08 July 2020
Available online 27 September 2020

Keywords:
Genetic algorithm;
Evolutionary Neural networks;
Reinforcement Learning;
Particle filters;
Markov decision processes;
POMDPs.

Abstract: This paper introduces novel concepts for accelerating learning in an off-policy reinforcement learning algorithm for Partially Observable Markov Decision Processes (POMDP) by leveraging multiple agents frame work. Reinforcement learning (RL) algorithm is considerably a slow but elegant approach to learning in an unknown environment. Although the action-value (Q-learning) is faster than the state-value, the rate of convergence to an optimal policy or maximum cumulative reward remains a constraint. Consequently, in an attempt to optimize the learning phase of an RL problem within POMD environment, we present two multi-agent learning paradigms: the multi-agent off-policy reinforcement learning and an ingenious GA (genetic Algorithm) approach for multi-agent offline learning using feedforward neural networks. At the end of the trainings (episodes and epochs) for reinforcement learning and genetic algorithm respectively, we compare the convergence rate for both algorithms with respect to creating the underlying MDPs for POMDP problems. Finally, we demonstrate the impact of layered resampling of Monte Carlo's particle filter for improving the belief state estimation accuracy with respect to ground truth within POMDP domains. Initial empirical results suggest practicable solutions.

1. INTRODUCTION

Recent advances in the field of artificial intelligence (AI) has unveiled a wide range of efficient algorithms [1] which if skillfully hybridized, could result in a plausible model for solving some of the problems in the field.

Since the introduction of value iteration algorithm for planning [2-5] in the 1970s, it has undergone a couple of refinement by numerous authors with an attempt to adapt it to solving more complex real world Problems. The combinational explosion of linear components (also referred to as the curse of dimensionality in some literature sources) in the value function is one of the major reasons that POMDPs are impractical for most applications [6-8]. Another related problem with value iteration is the exponential growth of distinct action-observation histories (also referred to as the curse of history). Some ingenious pruning methods have been used to ameliorate the problem but these pruning methods are in themselves computationally expensive to implement and only work for small finite horizon problems [9, 10].

Some better strategies have been implemented such as PBVI (Point Based Value Iteration) [3] which iteratively update a sub set of representative belief points. Another promising method implemented for both discrete and continuous belief states is the MCMDP (Monte Carlos Markov Decision Process) [6, 11]. This method attempts to map POMDPs directly to their underlying MDPs using Bayes Particle filter for belief updates.

On a parallel front, Reinforcement learning (RL) algorithm is considerably a slow but elegant approach to learning in an unknown environment. Although the action-value (Q-learning) is faster than the state-value, the rate of convergence to an optimal policy or maximum cumulative reward remains a constraint. However, RL has the advantage of learning an underlying MDP for both dynamic and stochastic environments [12-14].

In this paper, the authors via experiments, investigate the effect, impact or/and contributions of multi-agents to accelerating the rate (thereby shortening the duration) at which the utilities converge to an optimal policy for planning within POMDP environments. The agents leverage on the greedy strategy for online exploration-exploitation using off-policy model free algorithm [15]. We then compare this multi-agent RL model with an ingenious multi-agent framework equipped with a feedforward neural network which is optimized offline via an objective function (based on localization of the goal and absorbing nodes) using genetic algorithm. We then identify the promises and constraints of both paradigms and thus propose future recommendations.

Furthermore, because every POMDP can be mapped directly to its underlying MDP, we examine how an agent armed with a single range sensor could minimize the margin of error between an agent's belief state and its actual state via an ingenious resampling algorithm for the Monte Carlos particle filter [16-19]. The rational is to unveil (in the failure one or more sensors) a cost saving and relatively efficient approach to robot localization in POMDP environments. Empirical results show that this simple procedure quickly filters out outliers responsible for large errors in the initial approximate belief of an agent's state.

2. LITERATURE REVIEW

Hybrid Genetic algorithms Evolution computation is a field that includes genetic algorithm, genetic programming along with evolution techniques which capture the entire process of selection and mutation [20-22]. The biological model of natural selection and genetics form the basis on which these computational techniques are implemented. A class of 'random search algorithm' with theory firmly embedded in biological models of selection and evolution is referred to as *genetic algorithm* (GA). Given a clearly defined problem to be solved, a basic GA can be represented as a set of string of bits (chromosome) which could be decoded to represent a solution to the problem. Each chromosome is tested to see how good it is at solving the problem by assigning a fitness function to them [23, 24]. The probability of a chromosome being selected is proportional to its fitness. The higher the fitness score, the better the probability of chromosome being selected. A popular method of selection is the Roulette wheel selection. This iterative process unveils an ingenious paradigm for optimal path creation in maximizing the coverage of the search space when solving the multisource/target problem.

An evolutionary neural network is a hybridization of two powerful AI algorithms: the genetic algorithm and the artificial neural networks [25-27]. They are both biologically inspired and are often designed as feed forward ENNs when combined. This combination is achieved by evolving the weights in a fixed neural network while providing the network with a set of inputs [28-30].

2.1 REINFORCEMENT LEARNING

Reinforcement learning is the science of sequential decision making. For grid world agents, it is characterized by an agent's ability to maximize long term rewards leveraging on past experiences obtained via interaction with a stochastic environment. Because the environment is initially unknown to the agent, the agent has to surmount the challenge of handling the delicate balance between exploring and exploiting the environment while maximizing the expected long term reward. Consequently, RL agents usually combine online learning and planning simultaneously via policy optimization [31, 32].

The utilities of each state in RL are often referred to as state-value function. Analogous to the state-valued function is the action value function often referred to as Q-value function. The process of learning with Q-value functions is referred to as Q-learning [33].

$$Q(s_t,a)new = Q(s_t,a) + \propto \left(r_{t+1} + \gamma\, max_a\, Q(s_{t+1},a) - Q(s_t,a)\right) \quad (1)$$

where,

$Q(s_t,a)$ is the current value of the state under a specific action policy a;

r_{t+1} is the received reward;

$max_a\, Q(s_{t+1},a)$ is the maximum Q-value of the subsequent state under a specific action policy a;

$\propto$ is proportional to the learning rate weighted by γ the discount factor.

In this paper, we adopt a Q-learning RL for our implementation because it learns considerably faster than the state-value function. However, the reinforcement learning process is generally slow. Consequently, we attempt to accelerate the learning phase via the introduction of multi-agents.

2.2 MDP AND POMDPs

MDPs have a reputation for robotic navigations in a known environment. The environment is assumed to be Markovian (i.e., the effects of an action stochastically depends on the current state of the world and the executed action). Because the resulting state from the action is not deterministic,

the subsequent state of the agent may be unintended. Amidst the stochasticity, the robot must navigate from its current location to a goal location with the minimum possible steps. Thus, MDPs create a policy for every possible node in grid world that is fully observable and stochastic [3, 6, 11]. MDPs are usually defined as a tuple < S, A, T, R> where:

S- a set of environment states (which must encapsulate all relevant information for taking correct decisions – e.g., Map, exact location within map, state of the world (open or closed door).

A- all actions that the agent can execute. A simplified example would be UP, DOWN, LEFT, RIGHT;

T- the stochastic transition function T(S, A, S$^{'}$) = P(S$^{'}_{t+1}$ = S$_{o\ |}$ S$_t$ =s, A$_t$= a) – the probability of executing an action 'a' from state 'S' at time 't' and arriving at state S$^{'}$ at time 't+1';

R- the reward function which models the utility of the current state as well as the cost of taking a particular action R(S, a). A negative living reward (non-zero cost) is usually associated with grid world implementations.

In this paper, our simulation is based on planning problems which has a finite and discrete state and action space. The purpose of planning is to find a policy (set of optimal actions) that describes the agent's behavior in order to maximize the sum of expected rewards

$$U(s) = \sum_{t=0}^{\infty} E[\gamma^t \ R(S_t)] \quad (2)$$

where, γ is the discounted reward as 't' tends towards infinity $0 \leq \gamma < 1$. This keeps the solution bounded. However, since our horizon is finite, (i.e., has an absorbing or goal state) we set $\gamma = 1$

For every state S, we can compute a utility function with the following equation:

$$U(s) = R(s) + \gamma \sum_{s'} T(S, a, S')U(S') \quad (3)$$

The optimal utility for each state is given by the Bellman equation

$$U(s) = R(s) + \gamma \, Max_a \sum_{s'} T(S, a, S')U(S') \quad (4)$$

The optimal policy is given by the equation.

$$\pi^*(s) = argMax_a \sum_{s'} T(S, a, S')U(S') \quad (5)$$

In real world domains, most of the assumptions behind the implementation of MDPs fall apart because the agent cannot directly observe the state of the environment. POMDPs give us more efficient alternative to modeling real world problems via probability distribution over states also referred to a belief states. This is because the actual state of the world cannot be fully observed due to inaccurate sensor readings. Alternatively in POMDP environments, beliefs provide a sufficient statistic for the history thereby availing sufficient information for the optimal policy per state with the assumption that the underling MDP is also Markovian [3].

POMDPs therefore can be defined as belief-space MDP with the tuple < B, A, T, R$_B$ > such that:

-B is the set of possible states over beliefs over state S;

-A is the set of possible actions;

-T is the belief transition function T (B, a, B$^{'}_o$); representing the transition probability of starting a belief B, taking an action a, and arriving at a new belief state B$^{'}_o$.

-R$_B$ is the reward at each belief state.

Just like the MDP model, we define the Bellman update operator [9] for the Belief-Space MDP (POMDP) as:

$$U(b) = Max_a \left(R(b) + \gamma \sum_{b \in B'} T(b, a, b')U(b') \right) \quad (6)$$

Consequently, like MDPs the goal of POMDPs is to find the policy for action selection that maximizes the reward (b) .

2.3 PARTICLE FILTERS ALGORITHM

Particle filter is an elegant algorithm with the potential of mapping trajectory history into belief states which consequently aid agents to learn a mapping from belief states to action in POMDPs [34-36]. Particle filters are the implementations of recursive Bayesian filtering used for modeling non-Gaussian distributions [37, 38]. Using the motion and sensor observation model, the algorithm iteratively updates the belief-states via a sequence of prediction steps and correction steps usually referred to as belief updates [39, 40].

Predictor step is given by:

$$\overline{Bel}(x_t) = \int P(x_t \,|U_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}. \quad (7)$$

While the correction step is given by:

$$Bel(x_t) = \eta P(Z_t \ |x_t) \, \overline{Bel}(x_{t-1}) \quad (8)$$

Combining both equations, we get the Bayes particle filter equation as follows:

$$Bel(x_t) = \eta P(Z_t \ |x_t) \int P(x_t \,|U_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \quad (9)$$

where,

ɳ is the normalization factor

$Bel(x_t)$ is the belief of being in state x at time t.

$P(Z_t \ |x_t)$ is the probability of sensing $Z_t$given a state location x_t at time t.

U_t is the action or motion step at time t.

3. EXPERIMENTAL SETUP

The experiments can be divided into two sub sections: Section A and Section B.

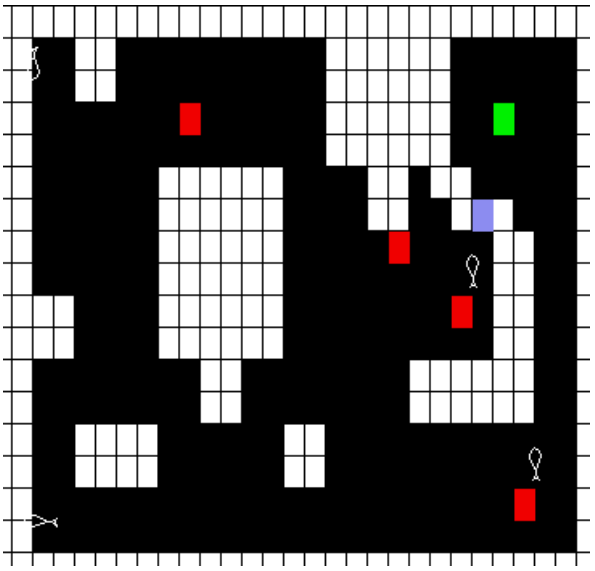


Figure – (1.0) Multi-agent RL environment with walls (white cells), absorbing states (red cells), dynamic door (blue cell) and goal node (green cell)

Section A

In this section (Section A), we show how the multi-agent Q-learning RL algorithm [41-45] converges quickly when compared with a single off-policy agent. It is important to note that learning algorithm creates an underlying MDP model for the grid world (Fig. 1) at convergence.

The first simulation had a single RL agent in a 30 X 20 grid world (Fig. 2) with obstacles (white cells), absorbing nodes (red cells), a single door (blue cell) which toggles (open/close) between episodes and a single goal node (in green). Below are the settings of the environment used for all simulations (single agent and multi-agent):

Living reward (R) = -0.1

Learning rate (α) = 0.1

Discount factor (γ) = 1.0

Maximum Reward: (R_{Max} goal node) = +5000

Punishment (Absorbing nodes) = -5000

In the second simulation (Fig. 3), three more agents were added to the single agent. In a deliberate attempt to investigate the significance of the addition of a single agent, we ran a third simulation with 5 agents (Fig 4.0).

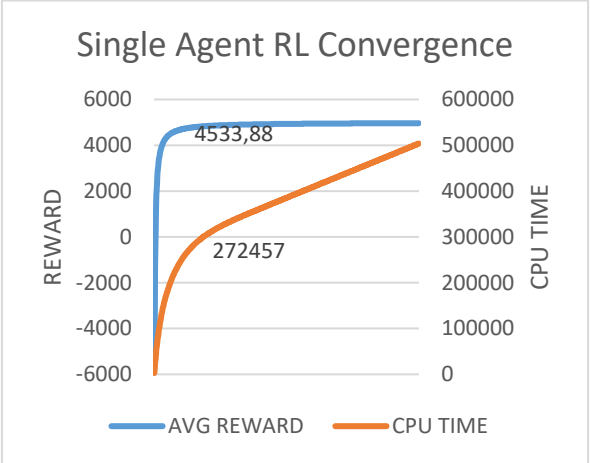


Figure 2 – Single agent reinforcement learning graph with respect to CPU-time

The results show a significant difference in the convergence rate. It is interesting to note that multi-agents displayed some emergent behaviors (outside the scope of this research) during the on-line training process while migrating the algorithm towards convergence.

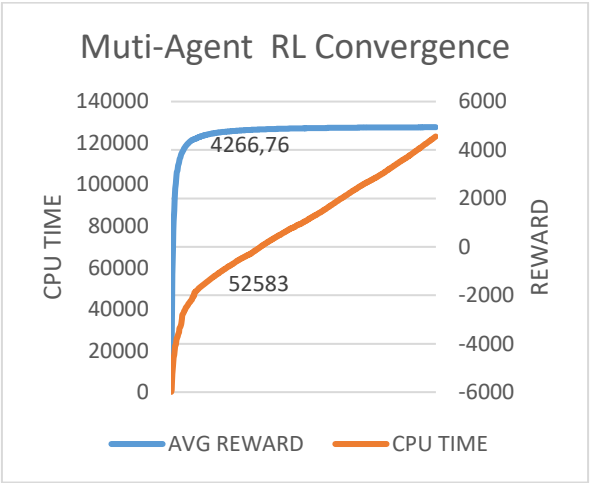


Figure 3 – Multi-agent (size of 4) reinforcement learning graph with respect to CPU-time

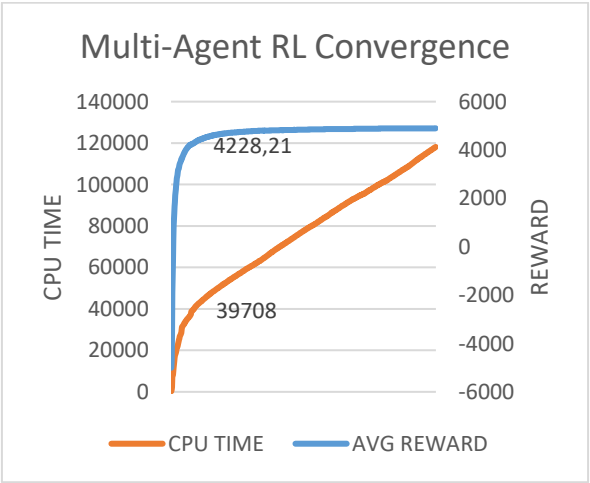


Figure 4 – Multi-agent (size of 5) reinforcement learning graph with respect to CPU-time

3.1 GENETIC ALGORITHM PARADIGM

In comparison, we simulate an alternate approach to creating an underlying MDP model for a grid world using multi-agents (4 agents) each equipped with feedforward neural networks, whose weights are optimized using genetic algorithm. The objective of function of these agents is to learn the model of the world via exploration. Training is done off-line via epochs over multiple generations. The fitness function for each generation of the multi-agents is given by:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} R(s)_{i,j} + \beta_{i,j} \qquad (10)$$

where,

R(s) are the living positive reward for each new explored state (i,j) in the grid world,

$\beta_{i,j}$ are extra rewards assigned to absorbing and goal nodes.

The iteration terminates after a predefined number of epochs or after a predefined minimum sum of rewards has been obtained. When the simulation terminates, it creates underlying MDP (Optimal policy) using dynamic programming with respect to the goal node. It is important to note that the entire learning procedure is considered to be off-line. Each epoch ran for a fixed duration (3750) CPU-time over 12 epochs (Fig. 5) before termination.

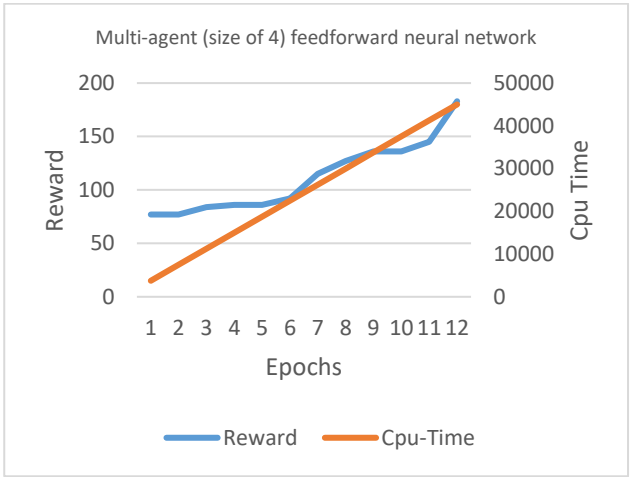


Figure 5 – Multi-agent (size of 4) feedforward neural network (with GA) learning graph with respect to CPU-time, and Epochs

Section B

In Section B, we simulate the planning phase for a single agent in a POMDP environment that leverages on the underlying MDP created in Section A. Our methodology incorporates the particle filter algorithm leveraging the roulette wheel selection for the resampling phase [46].

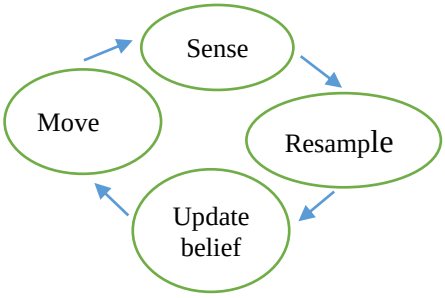


Figure 6 – Agent motion model for POMDPs

In our simulation, four sensor nodes are strategically placed at the edges of the grid world with which the agent is able to localize itself with respect to its belief update [47]. Gaussian noise was added to the sensor inputs. For simplicity, we discretized the agent's motion within the stochastic environment. The key idea is to efficiently map the belief state of the agent (particle filter averaged output) with the actual state of the agent. From (Fig. 6), the agent's policy is mapped directly to its belief which is based on the underlying MDP. Consequently, an accurate mapping would ultimately guide the agent to the goal node.

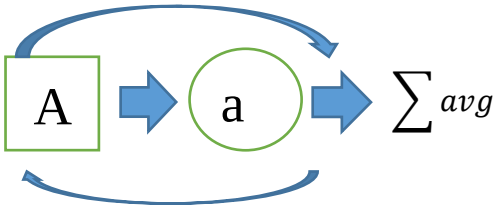


Figure 7 – Process flow diagram for traditional resampling and localization of belief state using particle filters. Capital 'A' (Initial Random sample), Lowercase 'a' (resampled)

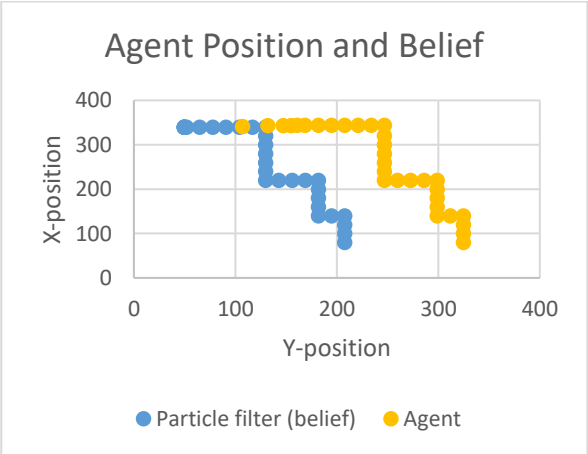


Figure 8 – Agent belief state (particle filter) and actual state transition from start position (upper left) to goal position (lower right) for traditional resampling

The resampling model depicted in (Fig. 7) is the traditional resampling model where particles are initialized randomly within the entire grid world [48] as depicted with the capital A. Thereafter, a new weighted sample based on important weights is produced (lower case a) via the roulette wheel selection algorithm. The x. y coordinates of the belief state are thereafter obtained by averaging the sum of the particles x. y coordinates. Fig. 8 shows the average result of this model. It is important to note that the agent motion model (Fig. 6) is iterated about five times with zero motion at the initialization phase before state transitions commence. The key idea is to minimize the error between the belief state and actual state before any transition begins. It is important to note that the initial state (position) of the agent in the world is unknown.

An improved model (Fig. 9) attempts to eliminate outliers resulting from the weighted samples by passing those samples through roulette wheel a second time to produce better weighted sample (Fig. 10) (lower case b) before averaging.

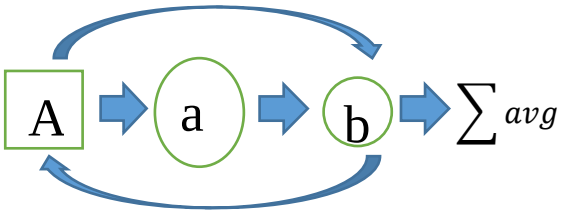


Figure 9 – Extended Process flow diagram for traditional resampling and localization of belief state using particle filters. Capital 'A' (Initial Random sample), Lowercase 'a, b' (resampled) with double phased resampling.

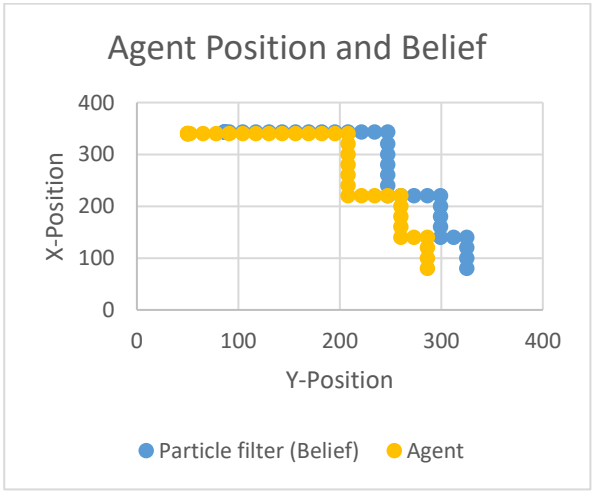


Figure 10 – Agent belief state (particle filter) and actual state transition from start position (upper left) to goal position (lower right) for double phased resampling

Introducing a third layer (Fig. 11) resampling produced even better results on the averages as shown in Fig. 12.

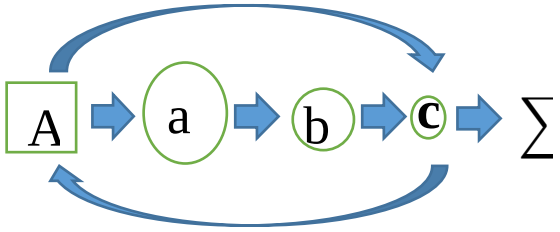


Fig. 11 – Extended Process flow diagram for traditional resampling and localization of belief state using particle filters. Capital 'A' (Initial Random sample), Lowercase 'a, b, c' (resampled) with triple phased resampling.

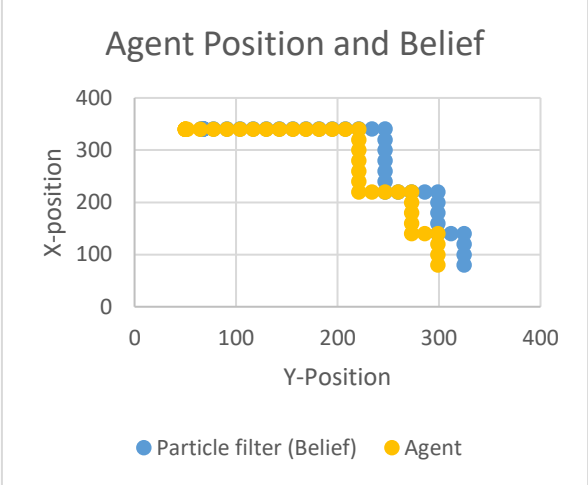


Figure 12 – Agent belief state (particle filter) and actual state transition from start position (upper left) to goal position (lower right) for triple phased resampling.

In our final model, we include a preprocessing phase with N (such that N =1000) number of particles randomly replicated 4 times in batches over the entire world as depicted in the A, B, C and D segments (Fig. 13).

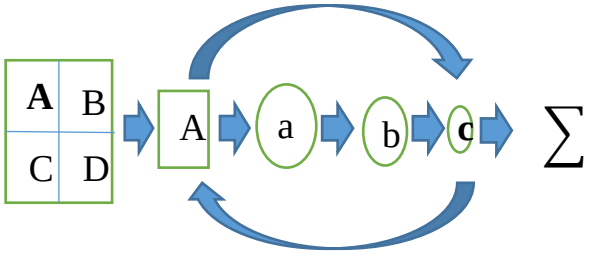


Figure 13 – Modified Process flow diagram for traditional resampling and localization of belief state using particle filters. Capital '(A, B, C D)' (Initial Random sample), Capital A (selected sample), Lowercase 'a, b, c' (resampled) with triple phased resampling.

The agent intuitively attracts the batch of particles with the highest probabilistic weights into the iterative phase, leaving behind other batches of N- particles. This procedure keeps the computational complexity simple while improving accuracy as shown in Fig. 14.

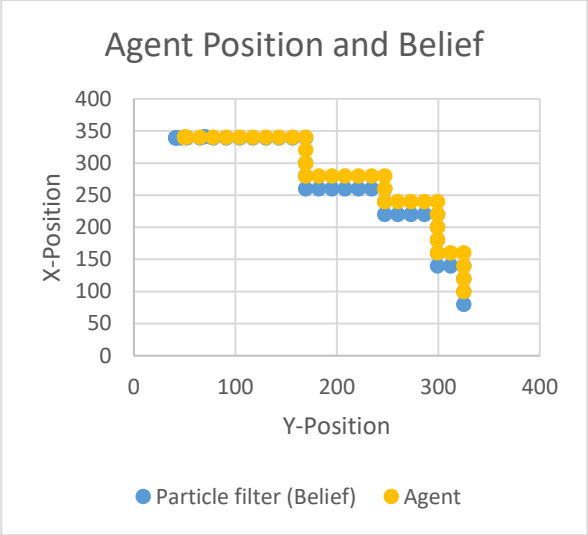


Figure 14 – Agent belief state (particle filter) and actual state transition from start position (upper left) to goal position (lower right) for preprocessed initialization with triple phased resampling.

This implementation drastically reduced the frequency of occurrence of false negatives (Fig. 15a the agent believes it is in a wall when it is actually not), and false positives (Fig. 15b the agent believes it is not in a wall, when it actually is) when observed over multiple runs. The final model maintained true positives (the agent's belief and actual state are approximately the same) over multiple runs as shown in Fig. 15c.

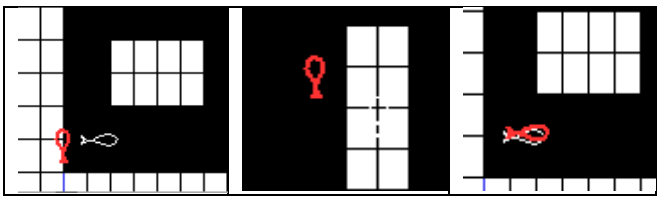


Figure 15 – (a) False negatives (the agent believes it's in a wall (belief in RED) when it's actually not), (b) False positives (the agent believes it's not in a wall, when it actually is). (c) True positives (agents position and belief are approximately same.

3.2. THE AMCL (ADAPTIVE MONTE CARLOS LOCALIZATION) APPROACH

The AMCL model is a relatively recent state of the art algorithm with which we compare our proposed localization algorithm. This algorithm randomly adjusts the number of free particles during the resampling phase based on their weights. By leveraging on the Kullback-Leibler divergence (KLD) algorithm [50, 51], the AMCL adapts a linear relationship to the number of particles in non-empty cells of the state space, and an upper bound on the number of resampled particles throughout the sense and move cycle [52]. Agent belief state and actual state transition from start position are shown in Fig. 16.

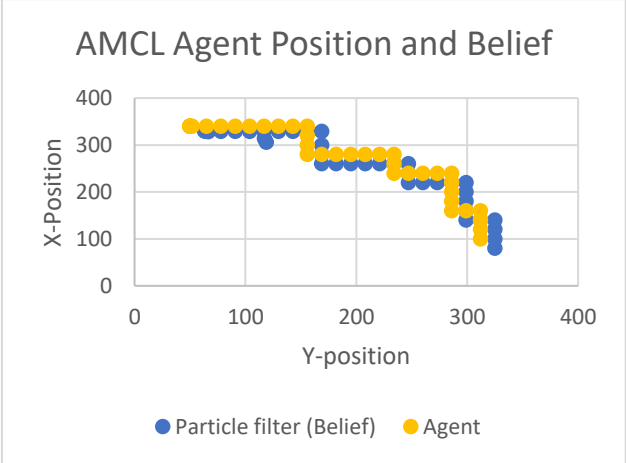


Figure 16 – Agent belief state (particle filter) and actual state transition from start position (upper left) to goal position (lower right) for the AMCL (KLD)

4. DISCUSSION OF RESULTS

We have obtained preliminary results for ongoing research in two phases: phase one for a typical learning problem and phase two for a complementary planning problem within a POMDP environment. In the first phase, we simulate learning of a POMDP environment using online, off-policy reinforcement Q-learning using both single and multi-agents. The rational is for the agents to learn optimal policy within a stochastic environment. The simulation results showed significant difference in CPU-time over episodes between the single and multi-agent frame work. The multi-agent (with a size of 4) converged much faster. With the addition of an extra agent, we witnessed even further improvement in CPU-time.

In contrast, we simulate an alternative off-line learning approach using feedforward neural networks for multiple agents (with a size of 4) whose weights were optimized using genetic algorithm over multiple epochs. This approach enables the agents learn the model of the world by localizing all absorbing states including the goal node and thereafter terminating with an optimal policy with respect to the goal node using dynamic programming [49]. This model converged faster than the Q-learning model however not without some drawbacks. The model is not naturally suited for dynamic environments (such as open/closed doors)

without a major modification to the algorithm which could impact computational complexity.

In the second phase (planning phase), results show how segmented initialization of N-particles combined with multi-layer resampling improved belief state accuracy with respect to ground truth for scenarios in which sensor fusion may be impracticable. Our proposed approach to the resampling phase revealed better accuracy when compared with the AMCL (KLD) algorithm. Consequently, the mapping of the POMDP to the underlying MDP was with relatively high fidelity.

5. CONCLUSION

In this paper, the authors compare two learning paradigms for POMDP problems and also contributed to the planning phase via a clever modification to the resampling stage of the particle filter algorithm. The proposed algorithms could be implemented in partially observable environments where a search and rescue operation may be required.

The multi-agent Q-learning showed more robustness for both static and dynamic environments, however it asymptotes relatively slower when compared with the multi-agent feedforward neural network counterpart. But then, the feedforward neural network offline learning paradigm is unable to adequately model dynamic environments.

The results from the grid world for state representation using multi-agent Q-learning showed that the increase in the number of agents, increases the rate of convergence. Though this may be true for the grid world with a computable finite state space, future research may reveal the veracity of this theory in more complex scenarios where states are represented using feature vectors.

Furthermore, we leverage on a classical resampling method (the roulette wheel) to demonstrate how an ingenious adaptation of the particle filter algorithm improved the belief state accuracy with respect to robot localization within POMDP environments.

ACKNOWLEDGMENT

This research is supported partially by South African National Research Foundation Grants (No. 112108&112142), South African National Research Foundation Incentive Grant (No. 114911), and ESKOM's Tertiary Education Support Program (TESP) of South Africa.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest regarding the publication of this paper.

6. REFERENCES

[1] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. Presa Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, issue 5, pp. 92:1–92:36, 2018.

[2] S. Obadan, Z. Wang, "A hybrid optimization approach for complex nonlinear objective functions," *International Journal of Computing*, vol. 17, issue 2, pp. 102-112, 2018.

[3] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," *Proceedings of the Int. Jnt. Conf. on Artificial Intelligence*, 2003, pp. 477–484.

[4] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," *J. Artificial Intelligence Research*, vol. 23, pp. 1–40, 2005.

[5] J. A. Bagnell, & J. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seoul, South Korea, 2001, pp. 1615-1620.

[6] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," *Proc. Neur. Inform. Process. Sys.*, Vancouver, Canada, 2010, pp. 1–9.

[7] C. Boutilier, & D. Poole, "Computing optimal policies for partially observable Markov decision processes using compact representations," *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1996, pp. 1168-1175.

[8] E. Hansen, & Z. Feng, "Dynamic programming for POMDPs using a factored state representation," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, Breckenridge, CO., 2000, pp. 130-139.

[9] S. Omidshafiei, A.A. Agha-Mohammadi, C. Amato, S.Y. Liu, J.P. How and J. Vian, "Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions," *The International Journal of Robotics Research*, vol. 36, issue 2, pp. 231–258, 2017.

[10] Z. Sunberg, and M. Kochenderfer, "POMCPOW: An online algorithm for POMDPs with continuous state, action, and observation spaces," *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018, pp. 1-9.

[11] H. Kurniawati and V. Yadav, "An online POMDP solver for uncertainty planning in dynamic environment," *Proceedings of the International Symposium of Robotics Research*, 2013, pp. 611-629.

[12] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al., "Value-decomposition networks for cooperative multi-agent learning," arXiv preprint arXiv:1706.05296, 2017.

[13] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. MIT Press Cambridge, 1998.

[14] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," arXiv:1709.06560, 2017.

[15] J. Schulman, P. Abbeel, and X. Chen, "Equivalence between policy gradients and soft Q-learning," arXiv preprint arXiv:1704.06440, 2017.

[16] J.-S. Gutmann, & D. Fox, "An experimental comparison of localization methods continued," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002, vol. 1, pp. 454-459.

[17] T. Patten, W. Martens, & R. Fitch, "Monte Carlo planning for active object classification," *Autonomous Robots*, vol. 42, pp. 391–421, 2018. https://doi.org/10.1007/s10514-017-9626-0.

[18] N. Cao, K. H. Low, & J. M. Dolan, "Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms," *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems AAMAS'13*, May 2013, pp. 7–14.

[19] G. Best, M. Forrai, R.R. Mettu and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1050-1057.

[20] M. Negnevitsky, *Artificial Intelligence: a Guide to Intelligent Systems*, second edition, Addison-Wesley, 2005.

[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[22] J. H. Holland, K. J. Holyoak, R. E. Nisbett and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*, MIT Press, 1986.

[23] L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann, 1987.

[24] R. Belew, L. Booker (eds.), *Genetic Algorithms*, Proceedings of the Fourth International Conference, Morgan Kaufmann, 1991.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[26] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation. Santa Fe Institute Studies in the Sciences of Complexity, Lecture Notes*, Addison-Wesley Longman Publ. Co., Inc., Reading, MA, 1991.

[27] C. Bishop, M. Svensen, & C. Williams, "GTM: the Generative Topographic Mapping," *Neural Computation*, vol. 10, issue 1, pp. 215-234, 1998.

[28] J. H. Holland, "Complex adaptive systems," The MIT Press on behalf of American Academy of Arts & Sciences, vol. 121, no. 1, pp. 17-30, Winter, 1992.

[29] J.R. Koza, *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, 1992

[30] P.W. Tsai, T.K. Dao, et al., "Robot path planning optimization based on multiobjective grey wolf optimizer," *Proceedings of the International Conference on Genetic and Evolutionary Computing*, Springer, 2016, pp. 166–173.

[31] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," arXiv preprint arXiv:1612.07307, 2016.

[32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 2000.

[33] H. Li, X. Liao, and L. Carin, "Multi-task reinforcement learning in partially observable stochastic environments," *Journal of Machine Learning Research*, vol. 10, pp. 1131-1186, 2009.

[34] T. Li, H. Fan, S. Sun, "Particle filtering: Theory, approach, and application for multitarget tracking," *Acta Autom. Sin.*, vol. 41, pp. 1981–2002, 2015.

[35] L. Martino, V. Elvira, G. Camps-Valls, "Group importance sampling for particle filtering and MCMC," arXiv, arXiv:1704.0277, 2017.

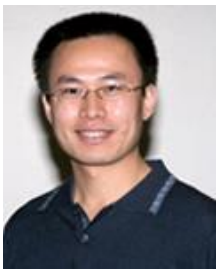
[36] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*, Cambridge, MA: MIT Press, 2005.

[37] G. Best, J. Faigl and R. Fitch, "Online planning for multirobot active perception with self-organising maps," *Autonomous Robots*,

vol. 42, pp. 715–738, 2018. DOI: 10.1007/s10514-017-9691-4.
[38] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. on Comput. Intell. and AI in Games*, vol. 4, issue 1, pp. 1–43, 2012.
[39] M. Toussaint, "Robot trajectory optimization using approximate inference," Proceedings of the ACM Int. Conf. on Machine Learning, 2009, pp. 1049–1056.
[40] Graeme, Best, et al., "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, issues 2-3, pp. 316-337, 2019.
[41] H. Li, H. Gao, T. Lv, and Y. Lu, "Deep q-learning based dynamic resource allocation for self-powered ultra-dense networks," Proceedings of the IEEE ICC Workshops, 2018, pp. 1–6.
[42] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," arXiv preprint arXiv:1810.07862, 2018.
[43] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, "An efficient deep reinforcement learning model for urban traffic control," arXiv preprint arXiv:1808.01876, 2018.
[44] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous networks," Proceedings of the IEEE International Conference GLOBECOM, Abu Dhabi, UAE, Dec. 2018, pp. 1–6.
[45] J. Perolat, J. Z. Leibo, V. Zambaldi, C. Beattie, K. Tuyls, and T. Graepel, "A multi-agent reinforcement learning model of common-pool resource appropriation," arXiv preprint arXiv:1707.06600, 2017.
[46] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000, vol. 1, pp. 256–263.
[47] W. Wang, J. Hao, Y. Wang, and M. Taylor, "Towards cooperation in sequential prisoner's dilemmas: a deep multiagent reinforcement learning approach," arXiv preprint arXiv:1803.00162, 2018.
[48] T. Li, J.M. Corchado, S. Sun, H. Fan, "Multi-EAP: Extended EAP for multi-estimate extraction for SMC-PHD filter," *Chin. J. Aeronaut*, vol. 30, pp. 368–379, 2017.
[49] S. Obadan, Z. Wang, "A multi-objective optimization approach to robot localization of single and multiple emission sources, *Procedia Manufacturing*, vol. 35, pp. 755-761, 2019.
[50] D. Sun, F. Geißer, and B. Nebel, "Towards effective localization in dynamic environments," *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, October 2016, pp. 4517–4523.
[51] S. Sun, T. Li, and T. P. Sattar, "Adapting sample size in particle filters through KLD-resampling," *Electronics Letters*, vol. 49, no. 12, pp. 740–742, 2013.
[52] G. Peng, W. Zheng, Z. Lu, J. Liao, L. Hu, G. Zhang, and D. He, "An improved AMCL algorithm based on laser scanning match in a complex and unstructured environment," *Complexity*, vol. 2018, article ID 2327637, 2018.

Samuel Obadan is a *Graduate student (Computer Science PHD), Masters (Information Systems), Bachelor (Computer Science). He studies at College of Science, Engineering and Technology, University of South Africa.*

His research interest area: Artificial intelligence, Optimization, Optimal Control, Fuzzy and/or Neural Network Control.



Prof. Zenghui Wang, *PhD (Control Theory and Control Engineering), Bachelor (Automatic Control). He works at College of Science, Engineering and Technology, University of South Africa. His research interest areas: Automatic Control, Adaptive Control, Predictive Control, Advanced PID Control, Optimal Control, Fuzzy and/or Neural Network Control, Fault Diagnosis and Fault Tolerant Control, Electrical Machine Control; Artificial Intelligence and Application: Particle Swarm Optimization, Genetic Algorithm, Energy (power system) Optimization, Multi-Objective Evolutionary Optimization, Deep learning Neural Network, Image/video Processing, Others: Industry 4.0, Encryption, Chaos, Internet of Things.*