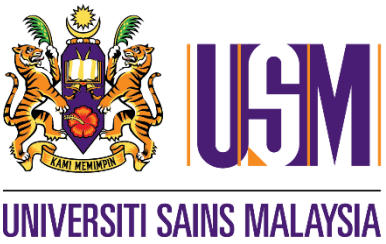


CPC251 - MACHINE LEARNING AND COMPUTATIONAL INTELLIGENCE

SEMESTER II ACADEMIC 2020/2021

ASSIGNMENT 2

GROUP CARDIO_4

Group Members	**Matric Number**
Loh Wan Teng	149104
Lim Zi Qiang	153116
Lim Phei San	152726
Pan Kwong Sing	152687

PREPARED FOR:

DR. MOHD NADHIR AD WAHAB

Reinforcement Learning

In this article, we will be discussing Improved Q-learning, Improved Deep Q Network and Deep Deterministic Policy Gradient where the first two are value-based model-free algorithms and the last one is policy-based model-free algorithm.

1. Improved Q-learning

Introduction (Sichkar, n.d.)

Traditional Q-learning implements ε-greedy policy which focuses on exploration in the begining and exploitation at the end of the learning process. In particular, the random Q-value initialisation due to unknown environment knowledge has reduced the efficiency to converge the Q-value. The improved Q-learning takes the trade-off between exploration and exploitation.

Algorithm and Techniques (Wang et al., 2020)

1. Q-value initialisation

   Q-value is initialised with Euclidean distance between the initial state and the terminal state to increase the "knowledge" of agent of the environment which reduces the "blindness" in the learning and exploration, to accelerate the training efficiency of Q-learning.

2. Action Selection Strategy

   Heuristic Searching Strategies is combined with greedy search to reduce the action randomness by searching an optimal solution for the current state. This assist the agent in to choose optimal solution during exploration instead of choosing an action blindly in the beginning of the learning process. Further with Boltzmann Exploration, actions are given probability to allow agent to balance exploration and exploitation over time that controls the probability spread of actions to be equally distributed in the beginning and sparsely distributed over the time.

3. Reward Function

   All actions that produce turns are given additional reward and actions that produce turns over 45^o is given relatively low rewards. In addition, actions that eagles the robot to move near obstacles will suffer a negative cost.

$$reward = reward + cost(turn(degree))$$

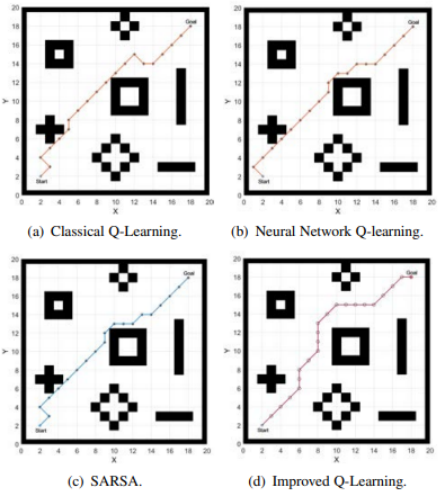
Result

Note: Classical or traditional Q-learning (CQL), neural network Q-learning (NNQL)

The performance of Improved Q-learning (IQL) has improved computational efficiency, smoothness and path safety compared to CQL. Although CQL has shorter path, it is very close to obstacles and many turns in the path generated by CQL which results in poor smoothing performance.

Table 1. Performance comparison between IQL, NNQL, SARSA and IQL based on the average result of 30 simulation experiments.

Evaluation Metrics	Experiment 1				Experiment 2			
	CQL	NNQL	SARSA	IQL	CQL	NNQL	SARSA	IQL
No. of Episodes	175.8	155.0	165.4	**141.1**	264.1	263.7	265.3	**124.4**
No. of 90° turns	1.6	1.5	2.0	**0.8**	1.4	1.4	1.1	**0.4**
No. of 45° turns	7.0	6.1	6.4	**8.0**	6.6	6.8	7.2	**8.8**
Length(unit)	16.2	16.0	16.0	**19.0**	20.0	20.0	20.0	**21.9**



(a) Classical Q-Learning. (b) Neural Network Q-learning. (c) SARSA. (d) Improved Q-Learning.

Fig. 4. World map 2 with mix obstacles.

2. Deep Q Network (Guan et al., 2021)

Introduction

Traditional Q-learning has its shortcomings in terms of small state dimensionality and slow Q-value convergence because it implements a table structure in storing the Q-value. Hence, Deep Q Network (DQN) is introduced with Q-function (not Q-table) which achieves the same result of mapping state-action pairs to a Q-value, $f(s,a)$. It uses neural network to represent the large state dimension as weights (Q-function) to estimate Q-function. Although the ordinary DQN algorithm can find a collision-free path to the target without knowledge of the environment the learning efficiency and convergence are slow.

Algorithm and Techniques

1. Adaptive exploration strategy

Adaptive exploration strategy uses an improved ε-greedy policy. In the original ε-greedy policy, an action is randomly selected under the probability of less than ε, and the action corresponding to the maximum action value Q is selected under the probability of greater than ε .

$$A(s) \leftarrow \begin{cases} argmax_a Q(s,a), 0 \le \theta \le \varepsilon \\ rand(A), \varepsilon < \theta < 1 \end{cases} \quad \text{where } \varepsilon \in [0,1]$$

This shows that ε is affecting the action taken by the agent. Hence, the improved ε-greedy policy feeds the ε with the reward based on the interaction between the agent and the environment.

$$\varepsilon = \begin{cases} \varepsilon_0 E^{|\Delta R_n|}, & \Delta R_n \le 0 \\ \varepsilon_0 + (1-\varepsilon_0) E^{|\Delta R_n|}, & \Delta R_n > 0 \end{cases}$$

where $\varepsilon \in [\varepsilon_{min}, \varepsilon_{max}]$, $E \in (0,1)$ and $\Delta R_n = R_n - R_{n-1}$ indicates difference between reward of n state and n-1 state .

This initiative is taking the feedback to "tell" the policy which actions yields a higher reward and the probability of taking that action, ε should be higher.

2. Heuristic reward function

Heuristic continuous reward is proposed to issue the problem of sparse rewards (eg. An agent has to be close enough to the target to receive positive reward).

$$R = \begin{cases} R_1, & \text{Reach the target point} \\ R_2, & \text{Hit an obstacle} \\ R_3, & \text{Intermediate state} \end{cases} \quad R = \begin{cases} R_q, & \text{Reach the target point} \\ R_o, & \text{Hit an obstacle} \\ R_{(a,S)} + R_{(n,S)}, & \text{Intermediate state} \end{cases}$$

where $R_{(a,S)}$ is intermediate reward and $R_{(n,S)}$ is the heuristic rewards.

Figure: sparse rewards (left), continuous rewards (right)

The heuristic reward function is defined as:

$$R = \begin{cases} R_q, & \text{Reach the target point} \\ R_o, & \text{Hit an obstacle} \\ R_{att} + \sum R_{req} + R_{(a,S)}, & \text{Intermediate state} \end{cases}$$

where R_{att} is the gravity reward function, R_{req} is the repulsion function when encountering an obstacle.

With the heuristic reward function, the agent can expect smaller expected reward when it gets closer to the obstacle (higher negative R_{req} penalty) and greater reward when it gets closer to the target (higher positive R_{att} reward).

Result

The improved DQN algorithm with heuristic reward and adaptive exploration strategy accelerates the learning efficiency, which results in faster convergence, faster path finding to find the optimal path.

Table 1. The corresponding reward and exploration strategy table of each algorithm

algorithm	reward	Explore strategies
Algorithm 1: Ordinary DQN	Sparse reward	e-greedy
Algorithm 2: DQN with adaptive exploration strategy	Sparse reward	Improve e-greedy
Algorithm 3: DQN with heuristic reward function	Continuous reward	e-greedy
Algorithm 4: DQN with heuristic reward and adaptive exploration strategy	Continuous reward	Improve e-greedy

Table 4. Simulation experiment results of 25*25 grid environment

Algorithm	The first round to find the target	Elapsed time (s)	Shortest path	3000 iterations total elapsed time (s)
Algorithm 1	71	25.96	48	949.59
Algorithm 2	55	22.41	48	931.55
Algorithm 3	49	12.06	48	529.56
Algorithm 4	38	10.10	48	533.33

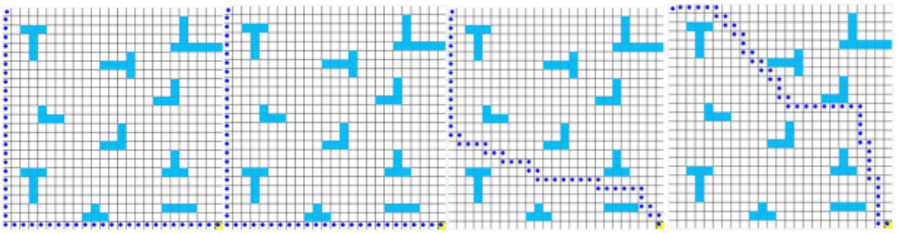


Figure 4. Visualization results of 25*25 grid environment pathfinding

3. Deep Deterministic Policy Gradient

Introduction (Jesus et al., 2019)

Deep Q Network can solve problems in complex environment with discrete action space (eg. one step in Chess) but not continuous action space (eg. turn 50^o of the wheel). Hence, Deep Deterministic Policy Gradient (DDPG) is proposed which uses an actor-critic network trained using Deep Policy Gradient approach to learn the action policy.

> Note:
> A policy is deterministic if it directly outputs the action.
> In Actor Critic Methods, the"Critic" estimates the value function (could be Q-value or V-value) and the "Actor" updates the policy distribution in the direction suggested by the "Critic"

Algorithm and Techniques (Paul & Vig, 2017)

The DDPG contains four networks: the actor network, μ and the critic network, Q, the target actor network, μ_{next} and the target critic network, Q_{next} .The actor network, μ takes the current state, s and output an action, a:

$$a = \mu(s) + N$$

where $\mu(s)$ is an actor function that deterministically maps states to continuous action values and N is the noises generated by the actor that promotes exploration to the agent (random N during initialisation).

Training Critic Network

At the same time, the reward, R is collected from the environment. Then, the critic network, Q takes the action, a and the current state, s, to generate the Q-value, Q. The next state, s_{next} is input into the target actor network, μ_{next} to get the next-state action, a_{next} and a_{next} together with the s_{next} is input to the target critic network to get next-state Q value, Q_{next} . The information obtained is stored in the replay memory in the form of tuple, $< s, a, R, s_{next} >$ which can be retrieved for Q_{next} to speed up the learning process.

The Q is trained and updated using the information obtained:

$$Q = R + \gamma Q_{next}$$

. To obtain the optimal policy, the goal is to minimise the critic loss between Q and Q_{next} to converge the Q-value:

$$\left|Q - (R + \gamma Q_{next})\right|$$

Note:
This is the reason why two critic networks is needed. When updating the Q-value, it is less likely for Q-value convergence when the Q-value comes from the same critic network. Hence, both the target networks will only be updated periodically with "soft" updates (not directly copy the weights of μ and Q):

$$\theta^{\mu}{}_{targ} \leftarrow \tau\theta^{\mu}{}_{targ} + (1-\tau)\theta^{\mu}$$
$$\theta^{Q}{}_{targ} \leftarrow \tau\theta^{Q}{}_{targ} + (1-\tau)\theta^{Q}$$

where θ is the weights of the target networks and τ is the parameter chosen on how similar the target network should be copied from the main networks.

Then, the process repeats until a finite amount of time, then the optimal policy is found.

Result

Unlike Value-based algorithms, policy-based algorithms is more efficient in terms of training time and large amount of actions because it directly finds the policy. Particularly, DDPG has shorter path planning time and lower number of path steps than the DQN algorithm.

Table 3. DDPG testing Performance

γ Value	% of Successful Attempts					Average No. of Steps				
	1500	2000	5000	7000	8500	1500	2000	5000	7000	8500
0.99	100%	100%	100%	100%	100%	1.95	2.92	2.42	2.21	1.98

Table 4. DQN testing Performance

γ Value	% of Successful Attempts					Average No. of Steps				
	1500	3500	5500	7500	10000	1500	3500	5500	7500	10000
0.2	31%	100%	81%	100%	100%	21.31	2.52	8.86	3.15	2.96
0.5	62%	64%	89%	97%	96%	16.19	12.35	6.11	4.7	6.12

Fuzzy Logic

1. Sugeno Fuzzy Inference (Takagi-Sugeno-Kang) (Kumar & Vamossy, 2018)

Introduction

The Sugeno Fuzzy Inference method or Takagi-Sugeno-Kang (TSK) breaks a large dataset into several smaller datasets and utilizes a collective mechanism to interact with them. The resulting component of this method is a linear combination of the input variables, and it leverages the collective knowledge of Collaborative Fuzzy Clustering (CFC) as input variables.

Algorithm and Technique Used

Inputs are fuzzified and applied to the fuzzy operator in this method. The output membership function is either constant or linear. An adaptive neuro-fuzzy inference system will be utilized to construct the membership functions of inputs and outputs as well as the rules for inputs for each input variable with their associated membership functions and output variables. According to the rules outlined in the sections before, crisp values will be transformed into fuzzy sets and then used by the inference engine. The Product-Sum-Gravity Fuzzy Reasoning Method is then used to define the fuzzy rules. For example,
$IF\ x_1\ is\ A_{11}\ and\ x_2\ is\ A_{21}\ and\ \dots\ and\ x_m\ is\ A_{m1}\ then\ y\ is\ B_1.$ After the conversion of the combination of updated sets to an output crisp value, the defuzzification module will denormalize the value. Because only the peaked value of modified fuzzy sets is taken into account, the height method is utilized as it is one of the simplest and most efficient defuzzification methods. The weighted total of the height values, which has been altered to yield a distinct output, determines the height of the fuzzy set. To minimize the learning error, the minimum of criterion function has to be found.

Results

Sugeno Fuzzy Inference method successfully avoids obstacles during robot navigation. This method can help the actual robot avoid obstacles in a real-world situation. The linear velocity of the real robot should be approximately that of the simulated robot in order to get a better result.

2. Mamdani Fuzzy Inference (Rawat et al., 2018)

Introduction

The Mamdani Fuzzy Inference method has been used to analyze mobile robot path planning. The Front Obstacle Distance (FOD), Back Obstacle Distance (BOD), Left Obstacle Distance (LOD), Right Obstacle Distance (LOD) and Heading Angle (HA) are the five inputs in this method, which are utilized in accordance with different rules to produce an output known as the Steering Angle (SA).

Algorithm and Technique Used

During fuzzification, all the inputs except HA are utilized triangular, trapezoidal and Gaussian membership functions and the fuzzy sets formed are Very_Near, Near, Medium, Far and Ver_Far. The input, HA, utilizes only the Gaussian membership function and the fuzzy sets for HA are Very_Negative, Negative, Zero, Positive and Very_Positive. The output, SA, is defined only by the Gaussian membership function and the fuzzy sets for it are similar to HA. Then, a knowledge database is formed by a number of predefined fuzzy rules for the fuzzy logic controllers. For example, Rule 1: IF (FOD is Very_Near) AND (ROD is Near) AND (BOD is Far) AND (LOD is Far) AND (HA is Positive) THEN (SA is Negative). During defuzzification, the centroid of area method is used to return a crisp output.

Results

The tables below show the comparison of the time taken and path length obtained during the experimental trials with the simulation results for 10 trials, respectively. The average deviation is found to be within 6%. In order to obtain a better navigational result, hybridizing can be done to modify this method in the future.

No. of Exercise	Time taken in simulation (TTS) from start to goal in milliseconds	Time taken in experiment (TTE) from start to goal in milliseconds	Deviation $\frac{(TTS-TTE)}{TTS} \times 100$	Average Deviation
1	4167	4390	5.351	
2	3967	4200	5.873	
3	3825	4060	6.143	
4	4051	4250	4.912	
5	4207	4410	4.825	
6	4590	4810	4.793	5.378
7	3945	4160	5.449	
8	4298	4560	6.095	
9	4454	4690	5.298	
10	4160	4370	5.048	

No. of Exercise	Path Length in Simulation (PLS) from start to goal in millimeters	Path Length in Experiment (PLE) from start to goal in millimeters	Deviation $\frac{(PLS-PLE)}{PLS} \times 100$	Average Deviation
1	2292	2385	4.057	
2	2182	2273	4.170	
3	2104	2230	5.988	
4	2228	2343	5.161	
5	2314	2416	4.407	
6	2525	2685	6.336	4.956
7	2170	2288	5.437	
8	2364	2465	4.272	
9	2450	2558	4.408	
10	2288	2410	5.332	

3. Tsukamoto Fuzzy Inference (Wulandari et al., 2018)

Introduction

In the Tsukamoto Fuzzy Inference method, the “if-then” rule is used for problem solving. Each result of the “if-then” rule will be represented as a fuzzy set with a single membership feature. There will be restrictions on obtaining crisp values when applying this method to fuzzification.

Algorithm and Technique Used

The three main parts of the Tsukamoto Fuzzy Inference method are fuzzification, inference and defuzzification. For fuzzification, this method transforms all the inputs into linguistic variables with a fuzzy set for each input. For example, the fuzzy set for the output variable, steering angle, will be Large_Left (LL), Middle_Left (ML), Small_Left (SL), Zero (Z), Small_Right (SR), Middle_Right (MR) and Large_Right (LR). The membership value for each fuzzy set will then be derived from its membership function and used in the inference phase. After that, rules are created by linking the input and output variables in an “if-then” format. Crisp input is received by the Tsukamoto Fuzzy Inference process and supplied to the rule knowledge base. The knowledge base will display fuzzy rules in “if-then” form. The membership value acquired through fuzzification after setting the rules to be utilized is used in this method to calculate the value of the prior membership. The centre of area method is the next defuzzification method employed in this method. Finally, the crisp output value will be determined.

Results

As a result, while using this method, there will be a limit to the crisp value produced. The Fuzzy Grid Partition, which may decide the number of fuzzy rules that make up the underlying model, can be utilized to bypass the limitation by introducing a new membership function. This method works better for situations involving making decisions, such as figuring out the optimal amount of production.

Evolutionary Algorithm and Swarm Intelligence

1. Multi-objective genetic algorithm (Alejandro et al., 2016)

   Introduction

   Mobile robot path search based on multi-objective genetic algorithm (MRPS-MOGA) is designed with a genetic algorithm with multiple objective functions. It performs meta-heuristic-based path finding for path optimization.

   Algorithm and Technique Used

   Paths are randomly produced as a population and the fitness function of each candidate path is determined based on the five objective functions such as path distance, smoothness of path, collision-free path, safety and travelling time.

   $$FF = \min D(A,B) + \max(\text{smoothness}) + CFP + \text{safety} + \min(TT)$$

   Path distance objective function is determined using Manhattan distance method to find the shortest path from starting point to the endpoint.

   $$D(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$

   Smoothness of path is measured based on energy consumption when robots travelling and encountering obstacles. It is measured in terms of joule.

   $$EC(t) = \frac{1}{\sum_{i=1}^{n} (D_A - O_i)}$$

   Multi-objective collision-free path is obtained by deriving a path which encounters the least obstacles and satisfying the chosen criteria for shortest distance and path smoothness. Safety is determined based on the minimum number of obstacles and collision robots interact. While travelling times is measured as follows:

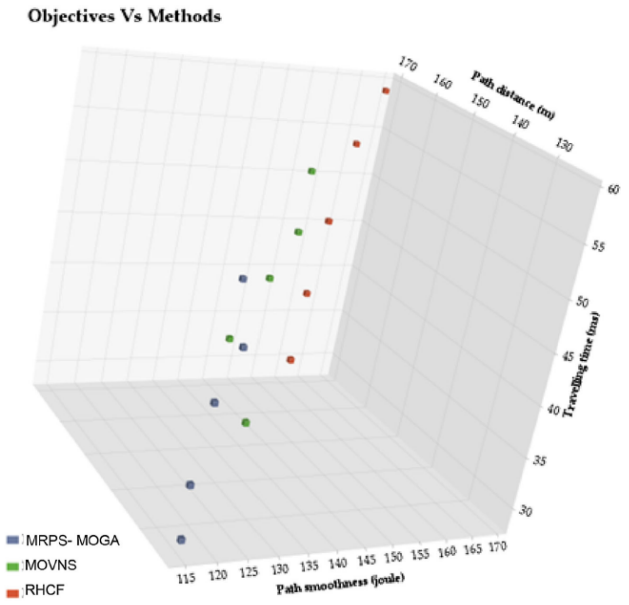
   $$TT = T_B - T_A$$

   If the path satisfies the fitness criterion, then the optimised path is selected, otherwise MRPS-MOGA applies three bio-inspired operators which are tournament selection, ring crossover, and adaptive bit string mutation. Tournament selection will select the best individual and ring crossover is carried out to obtain the best offspring. Finally, adaptive bit string mutation inverts the bit randomly to preserve diversity from one generation of a population of chromosomes to the next.

   Result

   The MRPS-MOGA is compared against the two existing methods namely MOVNS (Alejandro et al., 2016) and RHCF (Palmieri et al., 2016) based on various parameters. The shortest path is selected using MRPS-MOGA and the travelling time

is reduced by 31% and 42% as compared to existing MOVNS and RCHF respectively. The energy consumption is also decreased by using MRPS-MOGA by 15% and 31% as compared to other two methods. Time complexity is minimised by 20% and 33% as compared to other two methods. As a result, it is efficient using MRPS-MOGA in dynamic environments as it considers different performance parameters to select the optimal path with minimum time complexity. Moreover, it employs bio-inspired operators to verify the fitness criterion in order to achieve an obstacle-free path.



2. Ant Colony Algorithm with A* Heuristic Method (Dai et al., 2019)

Introduction

The traditional ant colony optimization (ACO) will require more time for searching path due to the lack of initial pheromone and the unapparent difference of the heuristic value. It leads to a slow convergence rate. While evaluation of A* algorithm can help improve the heuristic information which can accelerate the convergence speed during the path searching. The bending suppression operator is also implemented to improve the smoothness of the path.

Algorithm and technique used

A* algorithm (Duchoň et al., 2014) , which is developed based on the Dijkstra algorithm, is an efficient direct search method for finding the shortest path in a static road network. The heuristic cost of A* algorithm is expressed by the estimated function, f(n) as follows:

$$f(n) = g(n) + h(n)$$
$$h(n) = \left((n_x - g_x)^2 + (n_y - g_y)^2\right)^{1/2}$$
$$g(n) = \left((n_x - s_x)^2 + (n_y - s_y)^2\right)^{1/2}$$

where g(n) is the minimum cost from the source node to the current node, h(n) is the minimum cost from the current node to the destination node. n_x and n_y are the coordinates of the current node n. g_x and g_y are the coordinates of the target node g. s_x and s_y are the coordinates of initial node s.

The A* algorithm estimated function serves as heuristic information to find global optimal path while the bending suppression operator added is to reduce the number of bending times and the large cumulative turning angle. The improved heuristic information is as follows:

$$\eta_{ij}(t) = \frac{Q_2}{h(n) + g(n) + cost(bend)}$$
$$cost(bend) = \varphi \cdot turn + \psi \cdot thita$$

where Q_2 is constant more than 1, cost(bend) denotes as bending suppression operator, turn is number of turns from node n-1 to node n+1, thita is the angle between the line segment of node n-1 to node n and the line segment of node n t0 node n+1. ϕ is the coefficient converting turning times into grid length and ψ is the coefficient converting angle into grid length.

Result

The comparative analysis is carried out with the comparison of traditional colony algorithm, ant colony algorithm with nonuniform initial pheromone distribution (Zhao et al., 2016) and the improved ant colony algorithm with A* Heuristic method. The figure shows the comparison of three algorithms when the grid model is built into 30*30 grids. For shortest path length, the improved ant colony algorithm is basically the same as the algorithm (Zhao et al, 2016) but the number of bending times is reduced by 50% and 22% compared to both other algorithms respectively. When the number of grids increase to 40*40, both traditional ant colony algorithm and algorithm (Zhao et al., 2016) failed to search a global optimised path but the improved algorithm can still perform well.

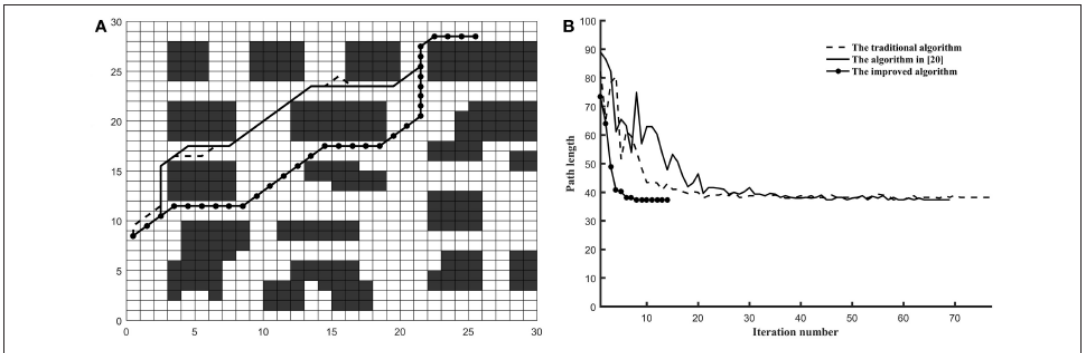


FIGURE 4 | The test results of three algorithms run on tunnel map. **(A)** Simulation results in 30*30 grid. **(B)** Convergence curve.

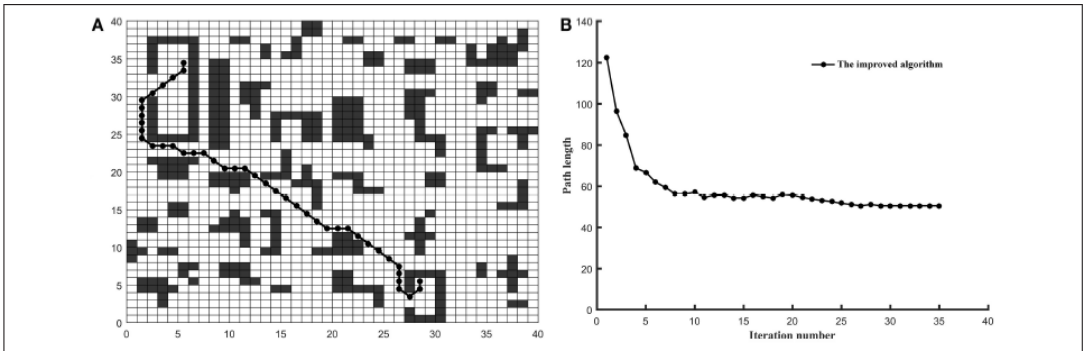


FIGURE 5 | The test results of three algorithms run on trough map. **(A)** Simulation results in 40*40 grid. **(B)** Convergence curve. (Other two algorithms is failed in trough map).

As a result, the improved ant colony algorithm has better adaptability and performance in complicated areas.

3. Particle Swarm Optimisation (PSO) with elite mean deviation induction strategy (Yang et al., 2021)

Introduction

Standard PSO algorithm will lose the diversity of the population in the later stage of the search, causing it not to be able to converge to the local extremum and leading to premature and slow convergence. To overcome this problem, an elite mean deviation induction strategy is introduced to increase the diversity of the populations.

Algorithm and technique used

Elite mean deviation distinguishes the diversity of particle distribution in the population by observing the dispersion degree of fitness function distribution. Individuals whose fitness is less than the average fitness are called elite individuals. When the population diversity is lost, the elite retaining strategy is implemented. For those individuals whose fitness is better than the average fitness, besides retaining the global optimal individuals, all other sub optimal elite individuals are initialised according to the probability. Hence, the population diversity increases. Moreover, particles with poor fitness are induced to leave and continue to search for particles which may contain global optimal solutions. Induction rules are as follows:

$$\mu_{id}(t+1)=\begin{cases}\eta\cdot k_1,\ fitness(x_i(t))\geq fitness(x_i(t-1))\\ \eta\cdot k_2,\ \ fitness(x_i(t))<fitness(x_i(t-1)\end{cases}$$

where μ is inducing factor; η is random number; k1 is positive number less than 1, k2 is positive number greater than 1. The update formula of particle velocity is as follows:

$$v_{id}^{(k+1)}=\omega v_{id}^{(k)}+c_1\varphi_1(p_{id}^{(k)}-x_{id}^{(k)})+c_2\varphi_2(p_{gd}^{(k)}-x_{id}^{(k)})+c_3\mu\varphi_3(p_{wd}^{(k)}-x_{id}^{(k)})$$

Result

Improved PSO algorithm with elite mean deviation induction strategy can obtain higher convergence accuracy and faster convergence speed than PSO algorithm as well as greater number of times to get the global optimal solution in test function.

Table 2. test results of improved algorithm and PSO algorithm on test function.

Test function		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
BPSO	Best	0.793e-52	-0.80432	0	0	96.7218	0	0	0.24779
	Mean	6.005e-51	-0.00431	2.3739	20.7321	89.3741	0	0	1.107e+02
	Worst	5.743e-51	-0.00327	19.4758	69.8432	90.7539	0	0	3.0789
	Iter	498	298	539	150	100	140	151	101
ICPSO	Best	0	-0.85456	0	0	96.8345	0	0	0.23815
	Mean	0	-0.37481	0	21.2178	151.3749	0	0	0.7932e+02
	Worst	0	-0.39543	0	75.3728	237.4123	0	0	1.37833
	Iter	291	317	405	78	231	24	21	59

Hybrid Intelligent System

1. Genetic Algorithms + Fuzzy Logic + Neural Networks + Expert Systems (O. Hachour, 2009)

Introduction

This Hybrid Intelligent System is constructed from the combination of Genetic Algorithm, Fuzzy Logic, Neural Networks and Expert Systems which is also known as the ES_FL_GA_NN system. It aims to assist Intelligent Autonomous Vehicles (IAV) to select the path that avoids obstacles to reach the predetermined target in an unknown environment. The intelligent system offers various benefits such as reducing communication overhead, improving runtime performance and providing design flexibility. This combination has proven to be superior to combinatorial optimization techniques and hence it can help the robot to understand the complex environment structure and reach the target without collisions.

Algorithm and Technique Used

Each intelligent system plays a different role in Intelligent Autonomous Vehicles (IAV) path planning. In the Genetic Algorithm approach, the path planning is represented as a square land while the path between two points is shown as the sequence of adjacent cells in the grid. Each grid in the square is assigned three states - free, occupied and unknown where free cell means no obstacles, occupied means contains one or more obstacles and other cells are labelled as unknown. The proposed algorithm starts by initiating the position at i by j grid and detecting the free, occupied and unknown cells within the grid. After that, a path from neighbouring free cells will be detected subject to two conditions. If the collection of free cells is continuous, the system will detect the move through it until the target is reached. The path is discontinuous, the system will change direction and continue on another path made of collection of free cells.

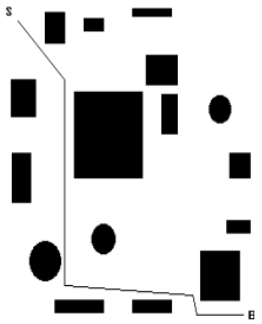
As for neuro-fuzzy system, it is composed of 4 neural layers of fuzzy system which aims to allow robots to determine the angle Cn for obstacle avoidance

formulation by using the output of neural network from the human expert knowledge captured in the fuzzy system. In the first layer, two angles, static danger degree for the n path and safety degree corresponding to the n path, [An, Bn] is fed into the corresponding membership functions of the second layer in order to perform fuzzification. The second and third layers further determine the direction and distance of the safety and danger instance around the robot. Finally, at layer 4, it computes the angle Cn corresponding to the output of the membership function and performs summation on the weights of the output layer.

To describe the flow in single sentence, it uses the best path generated from the biological genetic principle combined with the reasoning from the neuro-fuzzy system that captures the human expert knowledge to make choice of the best path direction that avoids dangerous obstacles.

Results

After testing the various unknown environments with static obstacles, the robot has successfully reached its target without collisions. The robots knows how to evolve and where it is situated from the target as shown in the figure on the left. Finally, it makes decision for choosing best to move and evolve without risk.



2. Swarm Intelligence + Fuzzy (Tao et al., 2021)

Introduction

As the Ant Colony Optimization method in path planning has inefficient convergence over long time, a global path planning method based on the improved Ant Colony Algorithm and Fuzzy Control is proposed. In this case, Fuzzy Control practices new pheromone distribution and updates approach to modify evaporation rate corresponding to stages.

Algorithm and Technique Used

Ant Colony Optimization algorithm mimics the behaviour of an ant colony which communicates and finds the global optimal path in an unknown environment based on pheromones. The concentration of pheromone is set to be inversely proportional to the length of the route. Hence, the path with a higher concentration of pheromones tends to be chosen by ants and the concentration will be even higher as more ants pass by. The theory of heuristic function η and tabu list $tabu_k$ are proposed to improve the efficiency of path planning. The tabu list ensures that the ants won't travel back to the nodes that have been recorded.

The ant moves from the current node i to an unvisited node j at time t based on the distance information from the ending point and the pheromone concentration on the path. If there is an unvisited node exists, the ant k will then determine the transition probability P_{ij}^{k}between nodes. P_{ij}^{k} .

The fuzzy controller consists of four parts which include fuzzification, rule base, fuzzy reasoning and defuzzification and uses two input variables $Iter$ and $Value$ which will output α and β. The output is managed by the convergence state of ant colony in three different stages to avoid being stuck in the local minimum. The fuzzy controller uses the Mamdani inference method in which it introduces the input and output rules by experience. The rules is set such that if $Iter$ is A AND Value B THEN α / β is C.

Results

	ACO		IACO	
	Best Path (m)	Convergence Iteration Times	Best Path (m)	Convergence Iteration Times
Map 1	30.3848	26	29.7990	25
Map 2	30.9706	22	30.3848	13
Map 3	32.1412	20	30.9706	16
Map 4	39.6985	62	31.7990	13

As shown in the table, the improved Ant Colony Optimization (IACO) algorithm is better than its original version in which it can converge faster and find a better optimum path in the three maps used. The fuzzy logic is proposed to control information heuristic factor α and expectation heuristic factor β. However, the IACO has a shortcoming as it needs to find a suitable fuzzy rule which has caused it to be computational expensive.

3. Fuzzy Logic + Reinforcement Learning (ZHU et al., n.d.)

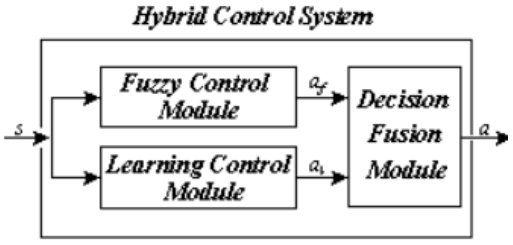
Introduction

The fuzzy modelization and planning is efficient but unable to perform consistent optimal control. Reinforcement learning can acquire the optimal control policy through on-line learning but it is slow in convergence. Hence, this paper proposes the combination of two methods to have better control performance over robot navigation in a dynamic environment.

Algorithm and Technique Used

The parallel structure hybrid intelligent system used in this paper consists of three modules named fuzzy control module, learning control module and decision fusion module. The parallel structure is built up from the two control modules which are based on the fuzzy approach and reinforcement learning respectively. The robot decides the subsequent step based on the sum of direction vectors generated separately by both control modules.

In the hybrid control system as shown in the figure below, both fuzzy control module and learning control module takes the environment state s and produce two distinct decision output a_f and a_l. a_f is the main element of the final decision due to fuzzy control module's efficiency while a_l is the sub-element that ensures the system's self-adaptability and the control policy's



global optimality. Before the a_f and a_l are passed into the decision fusion module each is assigned α and β as their weights respectively where α + β = 1. After that, the decision fusion module produces the weighted sum of the two decisions as the final output a according to this formula $a = \alpha a_f + \beta a_l$.

Within the fuzzy control module, the object's motion state in the dynamic environment is described as a fuzzy set that includes the information such as current position, speed and moving direction. Each object's motion state is defined in the membership function that is made up of the object's velocity vector V, vector D from the current position (x_c, y_c) with respect to (x, y) and R which represents the radius of the neighbouring area.

The learning control module takes multiple steps in one learning step during the learning process to discover the optimal control strategy. At first, it observes the environment state at time t as s_t. Then, based on the decision output a_t provided by the control policy, it moves to s_{t+1} while carrying the reinforcement signal, r_t. After that, the value of adjustment ε is calculated corresponding to the changes in the estimation of the state evaluation function and the control policy. The

Results

As shown on the right, the proposed hybrid intelligent system using fuzzy modelization and reinforcement learning demonstrates greater performance for efficiency and self-adaptability as compared to the fuzzy control alone. However, the system that provides a variety of knowledge representation is not fully studied.

Table 1 The data of the hybrid control experiment

Number of experiment	1	2	3	4	5	6	7	8	9	Average step numbers
Step numbers	309	289	366	344	535	340	315	349	554	377.9

Table 2 The data of the fuzzy control experiment

Number of experiment	1	2	3	4	5	6	7	8	9	Average step numbers
Step numbers	873	596	295	423	314	812	535	277	353	497.6

References

1. Alejandro, H.-P., Miguel A., V.-R., & Joaquín, F. (2016, Oct 1). *Applying the MOVNS (multi-objective variable neighbourhood search) algorithm to solve the path planning problem in mobile robotics.* ScienceDirect. Retrieved Jul 1, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0957417416301336?via%3Dihub#!
2. Dai, X., Long, S., Zhang, Z., & Gong, D. (2019, April 16). *Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method*. Front. Neurorobot. Retrieved June 30, 2022, from https://www.frontiersin.org/articles/10.3389/fnbot.2019.00015/full
3. Duchoň, F., Andrej, B., Kajan, M., Beňo, P., Florek, M., Tomáš, F., & Ladislav, J. (2014). *Path Planning with Modified a Star Algorithm for a Mobile Robot*. ScienceDirect. Retrieved Jul 3, 2022, from https://www.sciencedirect.com/science/article/pii/S187770581403149X?via%3Dihub#!

4. Guan, M., Yang, F. X., Jiao, J. C., & Chen, X. P. (2021). *Research on path planning of mobile robot based on improved Deep Q Network*. Journal of Physics: Conference Series. Retrieved July 1, 2022, from https://iopscience.iop.org/article/10.1088/1742-6596/1820/1/012024/pdf
5. Hachour, O. (2009). *The proposed hybrid intelligent system for path planning of Intelligent Autonomous Systems*. NAUN. https://www.naun.org/main/NAUN/mcs/19-134.pdf
6. Jesus, J. C., Bottega, J. A., Cuadros, M. A. S. L., & Gamarra, D. F. T. (2019, June 02). *Deep Deterministic Policy Gradient for Navigation of Mobile Robots in Simulated Environments*. 2019 19th International Conference on Advanced Robotics (ICAR). Retrieved July 1, 2022, from https://sci-hub.se/10.1109/icar46387.2019.8981638
7. Koubaa, A., Ammar, A., Alajlan, M., Sriti, M.-F., Chaari, I., Cheikhrouhou, O., Javed, Y., Bennaceur, H., & Trigui, S. (2018). *Robot Path Planning and Cooperation: Foundations, Algorithms and Experimentations*. Springer International Publishing. https://doi.org/10.1007/978-3-319-77042-0_1
8. Kumar, N., & Vamossy, Z. (2018, September). *Robot navigation with obstacle avoidance in unknown environment*. ResearchGate. Retrieved June 22, 2022, from https://www.researchgate.net/publication/328774686_Robot_navigation_with_obstacle_avoidance_in_unknown_environment
9. Muhammad, A., Ali, M. A. H., & Shanono, I. H. (2020). Path Planning Methods for Mobile Robots: A systematic and Bibliometric Review. 21. https://elektrika.utm.my/index.php/ELEKTRIKA_Journal/article/view/225/133
10. Palmieri, L., Rudenko, A., & Arras, K. O. (2016, Aug 24). *A Fast Random Walk Approach to Find Diverse Paths for Robot Navigation*. IEEE Xplore. Retrieved Jul 2, 2022, from https://ieeexplore.ieee.org/document/7549076
11. Paul, S., & Vig, L. (2017). *Deterministic Policy Gradient Based Robotic Path Planning with Continuous Action Spaces*. 2017 IEEE International Conference on Computer Vision Workshops. Retrieved July 1, 2022, from https://sci-hub.se/10.1109/iccvw.2017.91
12. Rawat, H., Parhi, D. R., Kumar, P. B., & Pandey, K. (2018, March). *Analysis and Investigation of Mamdani Fuzzy for Control and Navigation of Mobile Robot and Exploration of different AI techniques pertaining to Robot Navigation*. ResearchGate. Retrieved June 28, 2022, from https://www.researchgate.net/publication/324156551_Analysis_and_Investigation_of_Mamdani_Fuzzy_for_Control_and_Navigation_of_Mobile_Robot_and_Exploration_of_different_AI_techniques_pertaining_to_Robot_Navigation
13. Sichkar, V. N. (n.d.). *Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot*. 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). Retrieved July 1, 2022, from https://sci-hub.se/10.1109/icieam.2019.8742915
14. Suresh, K.S., Venkatesan, R., & Venugopal, S. (2022, June 23). *Mobile robot path planning using multi-objective genetic algorithm in industrial automation.* SpringerLink. Retrieved Jul 3, 2022, from https://link.springer.com/article/10.1007/s00500-022-07300-8#citeas
15. Tao, Y., Gao, H., Ren, F., Chen, C., Wang, T., Xiong, H., & Jiang, S. (2021, April 16). *A Mobile Service Robot Global Path Planning Method Based on Ant Colony Optimization and Fuzzy Control*. MDPI. https://www.mdpi.com/2076-3417/11/8/3605
16. Wang, J., Hirota, K., Wu, X., Dai, Y., & Jia, Z. (2020, October 31). *An Improved Q-Learning Algorithm for Mobile Robot Path Planning*. ISCIIA 2020. Retrieved July 1, 2022, from https://isciia2020.bit.edu.cn/docs/20201114081627901653.pdf

17. Wulandari, D. A. N., Prihatin, T., Prasetyo, A., & Merlina, N. (2018, August). *A Comparison Tsukamoto and Mamdani Methods in Fuzzy Inference System for Determining Nutritional Toddlers*. IEEE Xplore. Retrieved June 30, 2022, from https://ieeexplore.ieee.org/document/8674248
18. Yang, H., Jiang, L., & Wu, L. (2021, Sep 24). *Path planning of mobile robot based on particle swarm optimization algorithm*. IOPscience. Retrieved June 29, 2022, from https://iopscience.iop.org/article/10.1088/1742-6596/2093/1/012013
19. Zhang, Y., Sun, D., Wang, Y., & Wang, T. (2020, October 2). *The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning*. Frontiers. Retrieved July 1, 2022, from https://www.frontiersin.org/articles/10.3389/fnbot.2020.00063/full
20. Zhao, J., Cheng, D., & Hao, C. (2016, Sep 19). *An Improved Ant Colony Algorithm for Solving the Path Planning Problem of the Omnidirectional Mobile Vehicle*. Hindawi. Retrieved June 30, 2022, from https://www.hindawi.com/journals/mpe/2016/7672839/
21. ZHU, H., MASTORAKIS, N. E., & ZHUANG, X. D. (n.d.). *Hybrid Intelligent Control for Path Planning Based on Fuzzy Modelization and Reinforcement Learning*. ResearchGate. https://www.researchgate.net/profile/Nikos-Mastorakis/publication/308747899_Hybrid_Intelligent_Control_for_Path_Planning_Based_on_Fuzzy_Modelization_and_Reinforcement_Learning/links/57ee202908ae280dd0abe7e9/Hybrid-Intelligent-Control-for-Path-Planning-Base