WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Multi-Objective Deep Optimal Non-linear Reinforcement Support Learning

[1]Fagbohunmi, Griffin Siji, [2]Uchegbu Chinenye E.

[1]Department of Computer Engineering Abia State University, Uturu, Abia State, Nigeria
[2]Department of Electrical and Electronics Engineering Abia State University, Uturu, Abia State, Nigeria

ABSTRACT

This paper looks into the concept of Deep Optimal Non-linear Reinforcement Support Learning (DONLS) for the solution to high dimensional multi-objective decision problems needed where the proportion of the importance of each objective function cannot be determined in advance. In this problem scenario, the inputs from a high dimensional matrix are used based on their characteristic features to compute a set of convex range of matrix having the best solutions from the matrix set. Each matrix defines a dimension from the multi-objective decision problem for defining each objective function. The DONLS can be useful in such application areas such as solving complex problems like building construction, decision-making in engineering design and transportation. Two experiments were conducted using a test-bed and their result was used to validate the concept of the deep multi-objective reinforcement learning proposed in this paper. The results also show improvement over the neural network based solutions.

Keywords: Deep optimal non-linear support learning, deep multi-objective reinforcement learning,

1. Introduction

Today's world has witnessed tremendous increase in transportation networks, building design, bridge construction, automation of industrial processes just to mention but a few, in view of this, there exists complex decision in optimizing the multi-objective functions in many engineering applications, there is therefore a need for an efficient multi-objective decision making function to address these multifaceted problems. Advances in deep learning in recent times has led to the solution of many hitherto difficult reinforcement learning problems, these includes high dimension robotic control, [1 – 3], pattern recognition [4], riddle solving [5], learning environment for the Atari (LEA). The models stated in the above scenario have a single objective input, however in many real world scenario, the input will have multiple conflicting objective functions. It will be therefore required to obtain an optimal solution (policy) from these conflicting inputs. As an example, an automobile manufacturer may want to minimize the fuel consumption of a vehicle while at the same time maximizing its throttle (acceleration) of the vehicle [17]. Other examples include where most of the products and services required for a given set of population is provided for by an industrial activity, these includes production of drugs, clothing, renewable energy, production of vehicles and computers, just to mention but a few. In these scenarios, each company has many objectives such as maximising the number of customers, maximizing profits, minimizing raw material needed for productions etc. Al these objectives must be combined and an appropriate vector function representing an optimal policy for all objective functions computed.

These types of problems can be designed using multi-objective Markov decision process (MOMDP), and computed by employing multi-objective reinforcement learning. A multi-objective reinforcement learning (MORL) [18], is a model where an agent has many goals to achieve using different objective functions. At the end, all these goals will be maximized or minimized as the case may be to achieve an optimal solution for the entire system. In many multi-objective optimization problem, it is not always straight forward how to combine the trade-offs among the many objective function in order to arrive at an optimal solution to the entire system in advance, in order words there will not be a single optimal policy that satisfies all objective functions. It is therefore necessary to have a set of coverage that includes each of the many optimal solutions together with its value vector which defines its input matrix. The size of the vector is equal to the number of objective functions in the model.

It should be noted that even though deep learning technique has been applied in many applications involving the use of Markov Decision Process (MDP), it hasn't been extended to the Multi Objective Markov Decision Process (MOMDP) possibly because neural networks may not find optimal solutions to the many unresolved preferences in the model with its value vector sets. This paper will avoid the need for finding solution to these different vector plane in the MOMDP by using a concept known as the outer loop approach [19].

The outer loop concept does not attempt to find absolute solution to the different vector planes in the MOMDP, but rather train the neural network with the dataset contained in the MOMDP to find a coverage set of policies that approximates to their individual solution. By this an approximate solution to each of the resulting sets of vector planes and their corresponding unknown optimal solutions can be computed. At the end of the training, the neural

network will be able compute an approximate set of policies that represent a close to optimal solution for each vector plane by using a series of normalized singe objective function. Using this outer loop concept, the application of deep Q-learning networks that employs Multi-Objective Markov Decision Process can be done by using a technique known as optimistic linear support (OLS) [19, 20].

Hence multi-objective decision support problems (MODSP) can be solved using the OLS technique, which is a subset of the outer loop concept. OLS computes an approximate solution for each of the vector plane that represents the MOMDP and iteratively use it as a subroutine call to get an improved solution for each vector plane until an optimal solution of the entire vector planes in the MOMDP is found. An optimal solution for each vector plane is identified when the difference between two iterated solutions is less or equal to 0.005. This solution is referred to as the coverage set (CS) for each vector plane. Now, it should be noted that there will be as many optimal solutions as there are vector planes representing the MOMDP, however any one of this optimal solution can be selected to represent all other solution set as long as its use gives policy value vectors that correctly represent the composition of the entire vector planes in the MOMDP.

In this paper three new Deep Multi-objective Reinforcement learning algorithms will be proposed. The first algorithm will determine how the training of the neural network influences the OLS solution. The second algorithm determines how deep reinforcement learning can be made to conform to OLS solution standard. The third algorithm is a combination of OLS-compliant neural network solution and the OLS framework. This combination produces the Deep OLS learning system (DOLS).

From the empirical evaluation carried out in this paper, it was observed that DONLS can solve multi-objective decision support problem having a larger number of input datasets than the classical multi-objective reinforcement learning algorithms. The DONLS model was further improved by using the ability of the OLS framework in solving a range of single-objective problems especially when the range becomes similar as the range increases [21]. This scenario results in optimal solutions with similar value vectors. Deep Q-learning networks are capable of producing hidden features of a problem with respect to its function value. In this vein, parts of a network can be reused to obtain solution to a former single-objective problem so as to increase learning rate on subsequent objective problem. The combination of these abilities of Deep Q-learning networks is used in this paper to propose two novel algorithms, (i) Full reuse Deep OLS learning (FR-DOL) which is capable of reusing all parameter values of neural networks (ii) Partial reuse Deep OLS learning (PR-DOL) which is capable of reusing all but parameter values of the last layer of the neural network. It was shown empirically that reusing part of a neural network PR-DOL is more efficient than reusing all parameters of the neural network. PR-DOL is therefore able to improve the performance of multi-objective decision support system over the DOL without reuse and the FR-DOL.

2. Related Works

A Markov Decision Process (MDP) can be described as follows: An intelligent agent observes the current state in a single objective reinforcement model $cs \in S$ [22]. At each discrete time step t, it will choose an action $ac_t \in A$ with respect to a stochastic policy π, each action gives rise to a reward value $R(cs_t, ac_t) = r \in R$, after which it moves to a new state cs_{t+1}. The purpose of the agent is to maximize a reward over the range of transition made from its initial state to the current state, $R_t = r_t + \psi r_{t+1} + \psi^2 r_{t+2} + \psi^3 r_{t+3} + \ldots\ldots$, here r_t represents the reward obtained

Interests in Multi-objective reinforcement learning as recently gained importance in the field of machine learning. This was shown in the works of the researchers in [18, 25], where they did a survey on the applications of a multi-objective reinforcement learning, and the limitation of the single-objective reinforcement learning. The work of researchers in [26-28] looks into the modification of the operations of a single-objective reinforcement learning for the solution of a set of value vectors instead of the scalar value in the original model. However, the shortcoming in their work is that it cannot be applied to DONLS because the latter deals with vector 3-D space instead of a linear vector value used in their algorithm. In order words, their design was limited in scope as it could not deal with real life multi-objective decision support system needed for a back-propagation model used in this paper. In the work of the researchers in [29-31], they used neural network to design a multi-objective problem, but it is argued in this paper, that the use of Deep neural network can be extended to higher dimensional planes and give more accurate results especially when used with CCS.

A class of multi-objective reinforcement learning algorithms employ the heuristic policy search method which explore alternative policies with the hope of getting an optimum policy. Other designs were based on multi-objective revolutionary algorithms, these includes the following algorithms, MOEAS in [32, 33], and the Pareto local search in [34]. The performance of MOEAS is almost equal to that of neural network, but the performance of the evolutionary optimization of neural network is slow compared to back-propagation Deep neural network used in this paper.

It should be noted that apart from MORL, other algorithms are also based on OLS, however they can only be applied in a different scenario. Examples of this includes the OLSAR algorithm proposed by researchers in [21], it plans its environment based on a multi-objective partially observable Markov Decision processes, where reuse is made up of the alpha matrices which it uses to represent the multiple objective value function This is quite at variance with the proposal made in this paper where the alpha matrices represents the lower bound of a value function, which can be fully reused without affecting the exploration for an improved value function in future iterations

According to the researchers in [35], they proposed a variational OLS algorithm which was applied to the multi-objective coordination graphs while it reuses the reparameterization of the graphs obtained by a single objective variational interference techniques that was used as subroutine by the VOLS. It should be noted however, that the variational subroutine used in the variational OLS algorithm was not compliant to OLS unlike the DONLS technique used in this paper. The disadvantage of the variational OLS technique is that, separate policy evaluation step is required to retrieve the value vectors which makes it inefficient for the multi-objective deep reinforcement earning considered in this paper.

Problems involving Markov Decision Process are normally assumed to have a sequence of states i.e. source to destination and are modelled using a finite single-objective Markov decision process (MDP) by a tuple (E, D, G, Y, ψ). The Q-value of a policy ή is $Q^{ή}$(cs, a) = E[R_t cs_t = cs, a_t = a]. The function having optimal action is defined as Q*(cs, a) = $\max_{ή} Q^{ή}$(cs, a). The optimal value function obeys the Bellman optimality equation, given in equation 1.

$$Q^*(cs, a) = E_{s'}[R(cs, a) + \psi \max_{a'} Q^*(cs', a') \mid cs, a] \quad (1)$$

Unlike the Markov decision process which deals with a finite single-objective many state problems, Deep Q-learning networks (DQN) employs the use of neural networks to model finite multiple-objective multi-state problems. It has an additional parameter φ, which denotes the fact that it is multi-objective, this is denoted as Q(cs, a, φ). The optimization of DQN is obtained by minimizing the function given in equation 2.

$$Z_i(\varphi_i) = E_{cs, a, \lambda, cs'}[(x_i^{DQN} - Q(cs, a, \varphi_i))^2] \quad (2)$$

During every iteration, the maximum value of the DQN is given as x_i^{DQN} = r + λ$\max_{a'}$ Q(cs', a', φi) where φ_i represents the multi-objective parameters of the target network which is fixed for a given number of iterations needed to update the multi-objective network Q(cs, a, φ_i) using gradient descent. An action a is selected from one of the multi-objective functions denoted by Q(cs, a,φ_i) using the action selector which selects an action leading to the highest Q-value reward, with probability Pi, while otherwise it randomly choose any other action with probability 1 – Pi with the aim of learning more about the environment which most often lead to a long-term optimal Q-value rather than an immediate optimal value which may not overall lead to an optimal Q-value [23]. The aim of this random action selection is for the agent to learn more about the environment, thereby gathering more experience about its environment. This state of learning is usually referred to as the exploratory state of the agent, the exploration afford the agent to lower the variance between successive iterations while using former experience to decide between either using a greedy method to select the next action or to perform exploration [7].

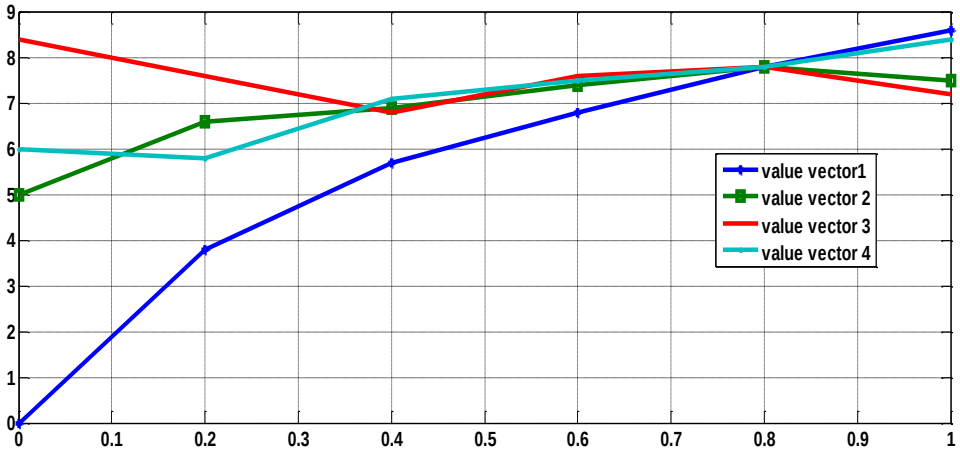

Figure 1 Three corner weight of Ut (w) when w has four value vectors in a 3-objective MOMDP

3. Multi-objective Markov Decision Process (MOMDP)

A multi-objective Markov Decision Process (MOMDP) is a type of Markov decision process where the reward function R(s, a) = $r_t \in \mathbf{R^k}$ that denotes a vector having k rewards, with one for each objective function [18]. In this paper, vectors will be represented in bold. A coverage set also known as set of policies which normally comprise of certain range of objective functions having at least an optimal policy amongst them is used as the solution for an MOMDP. The coverage set must have utility or scalable function fx. Scalable function maps every policy value vector, $\mathbf{V^{ñ}}$ to a scalar value. This paper focuses on a very common scenario where the scaling function is non-linear i.e. f($\mathbf{V^{ñ}}$, n) = $\frac{k}{u.V^{ñ}}$ here k is a real variable between 0 and 1, and **u** denotes a vector which computes the relative importance of the objective functions, in such a way that f($V^{ñ}$, n) represents the convex combination of the objective functions. The coverage set of this combination is referred to as the convex coverage set (CCS([18].

In the case of optimistic linear support (OLS), an outer loop approach is taken where the CCS is constructed incrementally by calculating a series of scaled single-objective functions for each linearly scaled MDP vectors **ñ**. In this way, DQN can be used to solve for single-objective Markov Decision Process. During every iteration, OLS decides on one policy by computing the scaled Markov decision process problem, while its vector value is summed with an intermediate approximated coverage set.

OLS is different from other outer loop methods because it uses a technique known as corner weight to select amongst the weights needed for the creation of the scaled instances, while it uses a technique known as estimated improvement to select mostly the subsequent corner weights. In order to know the corner weights to be selected, it will be first appropriate to know what scaled value function V_u^* is all about.

$V_s^*(u) = \max_{u\lambda \in s} \frac{k}{u.V^{ñ}}$ is a function of the linear scaled vector **u**, for a set of vector value $\boldsymbol{v}_w^*$**.(u)** where s contains four vector values as shown in figure 1. Here $\boldsymbol{V_s^*}$**(u)** will form a piecewise non-linear function in addition to a convex function which is made up of the higher surface of the scaled normalized

values for each of the vector values. Here the weights at the corner are represented by difference between the weights of the higher surface or upper end of the convex surface [24]. It should be noted here that the solution of OLS problem is the corner weight that maximizes a selection from the best higher bound from the difference between $\boldsymbol{V_s^*(u)}$ and the optimized value of the scaled normalized function i.e. $\boldsymbol{V_{ccs}^*(u) - V_s^*(u).}$ This represents the solution for the single-objective Markov Decision process normalized on the scalar value u.

The upper bound for the OLS can be calculated from the error on the upper bounds in relation to the optimized value of the normalized policy values from each previous state u of the series and the linear programs. Here the range of error encountered from two consecutive steps will improve as the OLS approximates the optimized value of the single-objective methods. On the other hand unlike in OLS where the initial values of the CCS is known in advance, this is not the same with reinforcement learning. The CCS value in reinforcement earning is not known in advance, therefore the upper bound for the optimized solution cannot be given based on the approximated model of deep Q-learning. In this kind of scenario, the difference between $\boldsymbol{V_s^*(u)}$ and the optimized value of the scaled normalized function i.e. $\boldsymbol{V_{ccs}^*(u) - V_s^*(u)}$ is used as the initial value to calculate the range. Here V_{ccs}^*(u) defines the highest normalized value of the CCS, as far as the values used for u in the previous state in the series have optima values for the convex

3.1 Methodology

This section describes the algorithm for the Deep Optimal Non-linear Support Learning (DONLS), It uses deep Q-learning with improvements from the OLS framework for multi-objective learning, In order to do this, the basic deep Optimal Non-linear (DONL) learning will first be proposed. Next the DONL framework was used to design a multi-objective learning while integrating the OLS framework. The DONLS algorithm was then improved upon using the Deep Optimal Non-linear Support learning with Partial reuse (DONLS-PR) and its full reuse counterpart (DONLS-FR). It should be realized that the DONLS, DONLS-PR and DONLS-FR uses single-objective functions which is usually integrated with the OLS framework as explained in section 3.1. Finally the DONLS algorithm is again improved on using Deep OLS.

3.2 Deep Optimal Non-linear Support Learning

Before one can use the OLS framework, two conditions must be satisfied, the first is that the problem should be a scaled version of a single-objective learning algorithm and secondly the problem must satisfy the requirements of OLS. Satisfying OLS requirements means that a vector valued Q-value **Q(s, a)** must be defined in the learning algorithm rather than a scalar Q-value for each transition. The vector valued Q-value must be a good approximation for the next corner weight of the OLS surface, i.e. each successive vector must give a good approximation for the next OLS surface. In order to meet these conditions, the neural network architectures was adjusted to produce a matrix of [C] x k (here, k represents the number of objectives while [C]represents the coverage set. The vector matrix so produced was used to train the transition of the agent.

The scaled deep Q-learning that uses the neural network architecture was first defined. The parameters needed to maximize the product of the corner weight of the OLS surface and the vector Q-values for a given transition was defined. The use of a scaled deep Q-learning function in OLS brought about the first algorithm.

3.3 Deep OLS Learning with Full Reuse (DONLS-FR) and Partial Reuse (DONLS-PR)

Even though DONLS is use to solve very large Multi-Objective Markov Decision Processes, where each objective is defined by a scalar value, thereby deriving a model for the full network, it may be difficult to use this same model where each objective is defined by a vector. However, a careful study shows that an optimal value vector of a scaled vector based multi-objective Markov Decision process occurs where the edge weights of two consecutive planes are close to one another, the optimal values for such planes will also be close [21]. The computation time for the optimal values of Multi-objective Markov decision process (MOMDP) can be reduced by the reuse of the neural network parameters of deep Q-networks. The deep Q-networks can extract the features of the MOMDP using reinforcement learning in order to obtain optimal values for it.

Algorithm 1 provided in this paper encompasses the three algorithms proposed in this paper. The algorithm is called Deep Optimal Non-linear Support Learning (DONLS). A closer look at the algorithm shows that each of the three algorithms were derived by using different values for the reuse parameter, for instance the reuse value for DONLS without reuse was set to 0, The reuse value for DONLS with full reuse was set to maximum value of 10, while the reuse value for DONLS with partial reuse was set to 5.

Algorithm 1 Deep DONLS Learning (having various levels of reuse)

1: function DONLS(n, u, template, reuse)

2: where, n – the (MOMDP) environment, u –threshold improvement,

3: template – specification of DQNLS architecture, reuse – represents type of reuse

4: T = null partial CSS

5: X = void list of explored corner weights

6: P = priority queue initialised with the maximum weights simplex with infinite priority

7: DQNLS Model = null table of DQNLS, depicted by the weight, x, for which it was learnt

8: while P is not void ∧ bt ≤ max_bt do

9: T = P.pop()

10: if reuse = ‘empty’ ∨ DONLS_Models is null then

11: model = an initialised random DONLS, from a pre-determined architecture template

12: else

13: model = copy_Closest_Model (X, DONLS_Models)

14: if reuse = ‘partial’ then initialise the last layer of model again with randomized weights

15: U, new_model = scaled Deep Q-Learning (n, T, model)

16: X = X ∪ x

17: if (∃x′) x′·U > max U ∈S x′·U then

18: X_{del} = X_{del} ∪ corner weights made redundant by U from P

19: X_{del} = X_{del} ∪ {x}

20: Isolate X_{del} from P

21: Isolate vectors from T which are no more optimal for any x after adding U

22: T_R = new CornerWeights (T, U)

23: T = T ∪ {U}

24: DONLS_Models[x] = new_model

25: for each x′ ∈ T_R do

26: if approximate Improvement (x′, X, T) > λ then

27: P.add(x′)

28: bt ++

29: return T, DONLS Models

In the case of DONLS-FR, that applies full deep Q-network reuse; the network will be trained for a new scaled weight x′, using the full network that was optimised for the previous x that is nearest to x′ in the sequence of the scaled weights which was generated by OLS. In the case of DONLS-PR the model employs partial deep Q-network reuse here the entire network is trained as in full reuse, however, only the last layer of the network will be randomly initialized in order to avoid sub-optimal values. In the case of DONLS with no reuse, no part of the network will be trained with previous values of the corner weights, in this case the entire network will be initialized at the commencement of each iteration

DONLS is capable of keeping track of partial CCS, corner weights wherein only a value vector will be included during each iteration this is shown in line four of algorithm 1. In order to locate these vectors, a scaled deep Q-learning was simulated for various corner weights. Corner weight which are not yet used is reserved in the priority queue, P and by the time they are used will be kept in a list X, this is shown in line 5 and 6 of algorithm 1. The priority queue is set up with the maximum weights and it also keeps track of the scaled weights which are arranged by approximate improvement. Now for the trained parameters in DONLS-PR/FR to be reused, the DONLS stores these parameters together with the corner weights x, this is used to define the DONLS models for partial reuse and full reuse.

The OLS framework for the DONLS proposed in this paper has a stack data structure, in which the scaled weight having the largest improvement at each iteration is added at the top of the stack while the rest corner weights are pushed down the stack, this is shown in line 9 of algorithm 1. Now as the corner weight with the highest value is popped on the stack, the DONLS reuses the weights it was trained with in previous iteration either fully or in parts according the DONLS reuse scheme employed. There is a function on line 13 of Algorithm 1 called copy_closest_Model, this is used to locate the corner weight closest to the current weight, in the case of DONLS-FR, the whole parameter value for the closest corner weight is copied, while in the case of DONS-PR, the corner weight of the last layer will be initialized again with random values as depicted in line 14 of Algorithm 1. Lastly, in the case of no reuse, where the reuse parameter is null, the entire network parameters will be reinitialized as shown in line 18 of Algorithm 1.

A closer examination on the various types of reuse employed on the scaled deep Q-learning was provided in section 3.1 As explained in section 3.1, the corner weight with the highest value in the OLS framework is added to the top of the DONLS stack, this is shown in line 9 of the algorithm. The output of the scaled deep Q-learning algorithm produces a vector value **U**, which represents the output of the trained policy of the DONLS model, this can be seen in line 15 of the algorithm. All corner weights selected from the number of iterations are pushed down the stack as depicted in line 16 of the algorithm. The order of these corner weights will be used to find out how the corner weights in future iterations will be prioritized. The criteria for

selecting future corner weights is based on the fact that, when there exist a weight vector **v** in the corner weight stack in which the scaled value is higher than any one from the vector in **T**, the value vector corresponding to the weight vector **v** is appended to T, while the next corner weight are computed based on the vector **T**. This corner weight will again be added to the top of the stack, this is shown in line 18 and 27 of the algorithm. However, the corner weight which correspond to **U** will be stored in the DONLS model only if it improves on **T,** otherwise it will not be stored.

The combination of vector **T** and the corner weight produces a new corner weight. This is used to improve on the solution of the multi-dimensional problem space to produce an optimized corner weights for edges in the system. The function new_Corner_Weights and estimated_improvements are used to compute new corner weights together with their estimated improvements. New corner weights are added to top of stack only if the value of their improvement is greater than λ as shown in lines 25 and 27 of the algorithm. In addition, unused corner weighs in the stack, which represents weighs not on the convex upper surface as computed with the new vector value are expunged from the stack as shown in lines 18 and 19 of the algorithm. This operation is repeated till all corner weights that cannot optimize the OLS framework is removed from the stack, after which the DONLS stops.

4. Results and analysis of Experiments

The performance of DONLS together with DONLS-PR and DONLS-FR will be evaluated in this section. Two multi-objective reinforcement learning solution referred to as hilly bus (HB) and deep river (DR) will be used to analyse the performance of these three variants of DONLS. The way in which DONLS together with DONLS-PR and DONLS-FR learn the true CSS by directly accessing the state of the problem is shown. The next step is the evaluation of how the technique used in this paper will perform in a larger network environment. Finally an evaluation of the performance of the various weight reuse variants are evaluated, for this purpose a pictorial version of the DST problem was designed where the input to the scaled deep Q-learning was bitmap

4.1 Experimental Setup

The DQN setup used by researchers in [7] was used to evaluate both real and pictorial version of the DONLS. Experience replay and a given network was used to train the network. Best fit exploitation / random exploration policy with annealing was employed, using a range of between -2 to 0.01 in the first 2500 and 3500 transitions respectively, while training is continued for equal number of transitions. The discounted value used for the reinforcement learning was -0.05, while the used network was initialized every 150 transitions. In order to make the learning converge, parallel transitions of 40 batches was used. These parameters were optimized using Adam with a learning rate of 10^{-4}. For each of the experiment, an average of 5 runs was used for the experimental analysis

In the case of the real state model of DONLS, an MLP architecture was used with 2 hidden layers of 150 neurons with rectified linear unit activations. In order to evaluate the 4 X 12 X 10 image inputs of deep river, two convoluted layers of 20 X 4 X 4 and 40 X 4 X 4 was used in a fully connected layer

4.2 Multi-objective Hilly Bus

In order to show that DONLS together with DONLS-PR and DONLS-FR can be trained with CCS, it was first tested on the multi-objective hilly bus problem (HB). Hilly bus is a variant of mountain car problem used in [22]. It should be noted that in a single objective hilly bus problem, a bus situated in a valley between two hills are controlled by an agent, while attempting to make the bus get to the right side of the top of the hill. It should be noted here that the bus is constrained in engine power. Also the intelligent agent must be capable of oscillating the bus between the two hills until such a time when the bua has gained adequate inertia to enable it get to its goal.

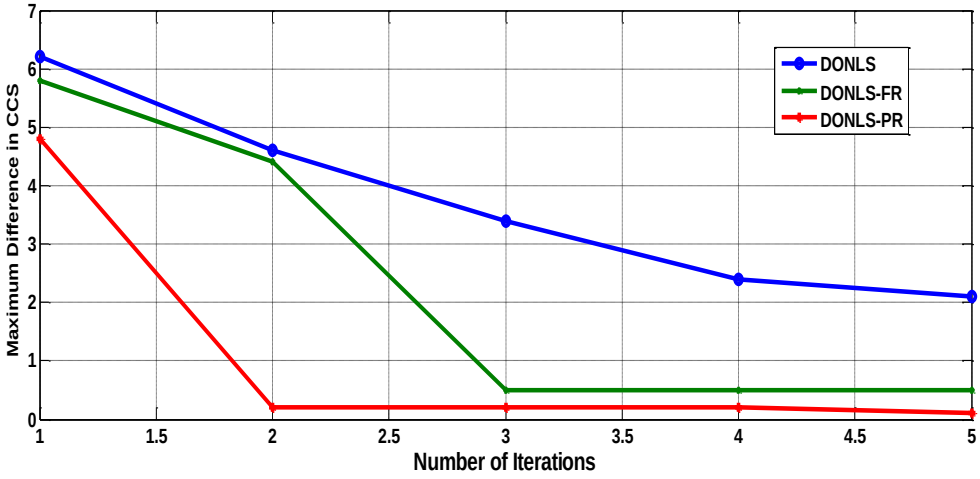

Figure 2 Mean value of raw HB version

In the hilly bus reinforcement learning problem, the singe-objective variant reward is -1, for all transitions that does not directly lead to the expected goal, while it is 0 when the transition leads to the desired goal. In the multi-objective variant which is considered in this paper, many factors will be considered in each transition towards to desired gaol. One of such is the fuel consumption required for each transition, this parameter is directly proportional to the force employed by the car. The Hilly Bus (HB) contains just two value vectors in the convex coverage set, hence its solution is not difficult.

Original Version: In this experiment, an evaluation of the DONLS architecture in the network area of HB was performed. Here the intelligent agent has direct link with the state (st) of the agent. The same neural network architecture with the deep river was employed. There was a different CCS used in comparison to the HB, for the deep river architecture, its CCS was derived from a Q-table algorithm from which the maximum CCS error was computed. It should be noted here that the Q-table presents the data which was used to compute the optimum value of the CCS. This is quite different from the case of HB where the CCS was randomly selected from the stack of corner weights. A closer look at figure 2 shows that the three variants of DONLS algorithm produce very close results with the HB problem where it can be observed that DONS-PR has the lowest error rate of 0.2 as compared to 0.6 and 1.9 for DONLS-FR and DONLS respectively. It took DONLS-PR just two iterations to learn a good approximation to the CCS while the number of iterations to achieve the same effect for DONLS-FR and DONLS was 3 and 4 iterations respectively, after which only little improvements were made to the vectors as witnessed from the graph. The performance of the three algorithms were equally impressive because the first two, three and four iterations in each case produced the optimal policy while the corner weights in each variants were very different from one another, so the value of the corner weights was not related to the number of iterations required to get the optimal policy

4.3 The Deep River Architecture

In order to validate the scalability of the DONLS algorithm proposed in this paper, the three algorithms were tested on a larger CCS. The deep river problem benchmark for MORL was adapted into the algorithm 1. In deep river (DR), the intelligent agent controls a submarine in search of desired goods in a 12 X 12 grid. The state of the agent (st) is made up of its current coordinate (x,y). In the experimental setup, the grid is assumed to have 12 desired goods whose rewards is directly proportional to their distance from the origin. The action spaces of the intelligent agent comprises of its search in the four cardinal points of the network space, this is shown in figure 3.

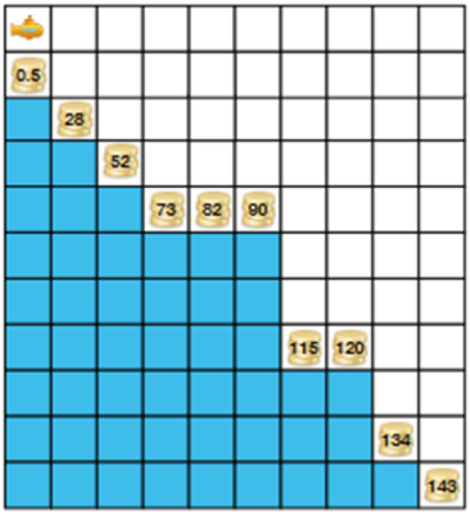

Figure 3 Image Map of DST

The transition of the intelligent agent results in either of two objectives, i.e. the desired goods is found or not. If the desired goods are found the reward is 100, otherwise, the reward is zero with a penalty of -1 for each transition that does not lead to the desired goods. In order to learn the CCS, in place of a Pareto front as used by researchers in [25], the values of the desired goods in the experiment was modelled in such a way that the best policy for getting to a desired good is always in the CCS. For easy evaluation the rewards of both objectives i.e. reaching or not reaching the goal were normalised between 0 and 1

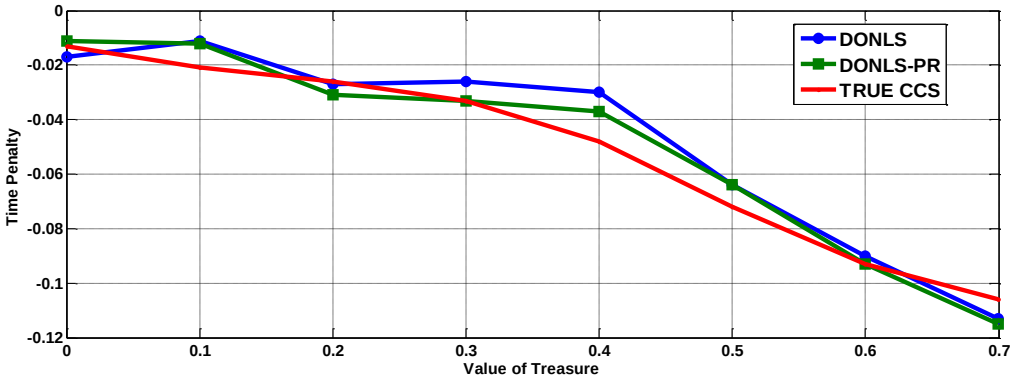

Figure 4 Graph of CCS after 3000 transitions for the raw version of DST

Original Version: in this scenario, the algorithms were evaluated where it was assumed that the intelligent agent had direct link to all the states that can be transited to in the network. Therefore, a backpropagation neural network was used to evaluate the maximum error using a scaled value in comparison with the actual CCS. The actual CSS value can be computed using the Multi-objective Markov Decision Process algorithm. The maximum error derived from the backpropagation neural network is denoted as Max_{CCS} error. Figure 4 shows a pictorial representation of the measured difference between the analytical CCS and the neural network generated CCS. From figure 5, it was observed that DONLS without reuse has the highest error of 0.09 while DONLS-PR has the lowest error of 0.055. This result contradicts the expectation held earlier that the CCS having direct link to the agent's states causes redundancy in the feature extraction and reusability of the network. It was also observed from the backpropagation neural network used that whenever the initialization model of the DONLS-FR corresponds to the optimal policy, its Max_{ccs} error was greatly increased in comparison to DONLS-PR. It was therefore concluded that the three variants of the DONLS algorithms proposed in this paper will give a good approximation to CCS solution, while the reuse option makes learning the network more accurate.

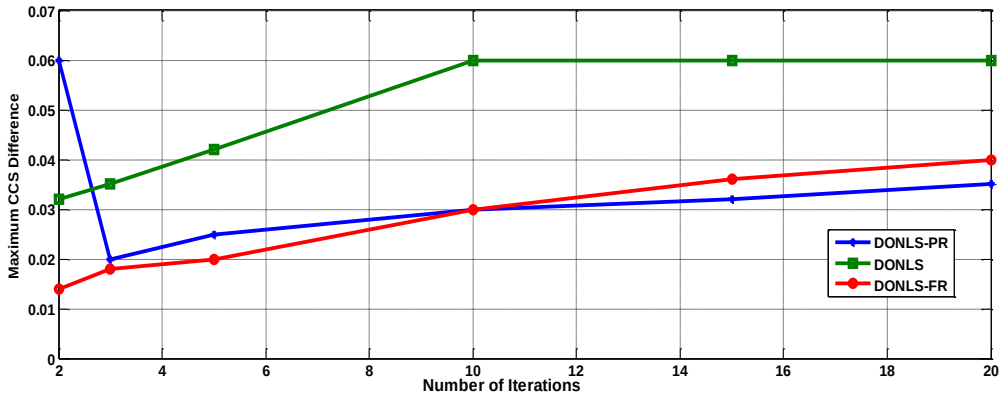

Figure 5 Raw version of DONLS

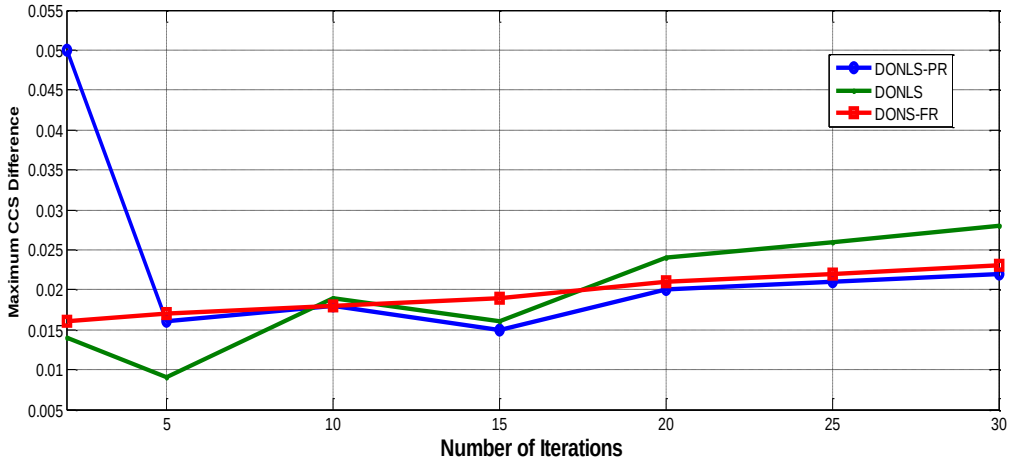

Figure 6 Image version of DONLS

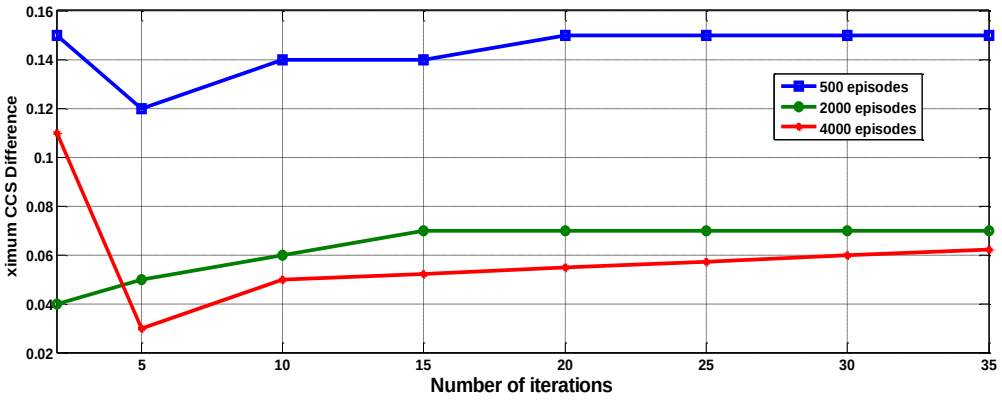

Figure 7 Accuracy Vs Episodes of DONLS

Pictorial representation: The deep convolutional architecture used for the pictorial representation of the DONLS has similar characteristics with the original version as it was able to give a good approximation for the CCS, with a very high degree of accuracy. This was shown in figure 6 where the reuse variants of the DONLS has improved performance over the DONLS without reuse It was also observed that the DONLS-PR has the highest stability among the three variants as its value did not result in high range value, its CCS range difference was 0.022 as compared to 0.028 in DONLS. The reason for this is due to the fact that with time, the network was able to learn the encoded space of the agent's states. The understanding of the encoded state spaces in turn enables an accurate build-up of the Q-values. From figure 6 where the minimum rage of CCS difference was analysed, DONLS-PR has the best performance of 0.06 as compared to 0.15 in DONLS, this can be attributed to its ability to reset the last layer in the Markov sequence. In the experiments conducted, the DONLS algorithm was built to train the multi-objective decision making problem so as to obtain an optimal way of traversing the network to achieve maximum reward. A closer look at the graph of the three variants of the DONLS show that DONLS-PR performs best.

Lastly, the importance of the number of times in which the algorithm was made to iterate before concluding on the maximum CCS error was investigated. A closer look at figure 7 shows that the error is dependent on the number of iterations performed (number of episodes). In the case of DONLS-PR, a small number of training episodes was not able to give adequate accuracy for building the CCS. As can be seen in figure 7, the CCS error progressively decreases up to 4000 episodes, but begins to experience diminishing returns from then, as the episodes gets up to 9500 episodes the network becomes over fitted, which leads to decreasing performance.

5. Conclusion

This paper proposed three novel algorithms using deep Q-learning for a multi-objective reinforcement learning problem. The algorithms were based on the optimistic linear support framework. Each of the techniques proposed was trained to learn a policy and corresponding value vector in each iteration towards the attainment of an optimized solution. The DONLS technique was further extended to get the best out of the neural networks. This was made possible by introducing the full reuse DONLS, (DONLS-FR), and the partial reuse DONLS (DONS-PR). This was used between the different iteration for the three techniques to enable faster learning.

From the readings obtained from the experiments conducted in this paper, it was shown that DONLS-PR performed better with large inputs when compared to DONLS and other compared algorithms. The DONLS algorithm was also able to learn the background CCS faster and was able to compute the value faster and with high premium. From the experiments conducted, DONLS-PR has better performance compared to both DONLS and DONLS-FR, this implies that (i) reuse of iterated parameters were important and (ii) performing partial reuse has the advantage of preventing the model from non-convergence over the full reuse counterpart especially for policies that are optimal for a previous weights. The future direction of this work will include fast stopping technique with the aim of optimizing the model presented here to compute accurate requirements of OLS, with reduction in the number of episodes.

References

[1] Levine S., Finn C., Darrell T., and Abbeel P. (2018) End-to-end training of deep visuomotor policies. arXiv preprint arXiv:

[2] Assael Y. M, Wahlström N, Schön T. B, and Deisenroth M. P. (2019) Data-efficient learning of feedback policies from image pixels using deep dynamical models. NIPS Deep Reinforcement Learning Workshop.

[3] Watter M, Springenberg J. T, Boedecker J, and Riedmiller M. A. (2020) Embed to control: A locally linear latent dynamics model for control from raw images. In NIPS Journal Vol 3 pp 345=356.

[4] Ba J, Mnih V, and Kavukcuoglu K. (2021) Multiple object recognition with visual attention. In ICLR journal, Vol 12 pp 465 - 478.

[5] Foerster J. N, Assael Y, M, and Whiteson S. (2020) Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:

[6] Mnih V, Kavukcuoglu Silver K. D, Rusu A. A, Veness J, Bellemare M. G, Graves A, Riedmiller M, Fidjeland A. K, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, and Hassabis D. (2020) Human-level control through deep reinforcement learning. Nature, in IJSP journal Vol 5 pp 529–533.

[7] Tesauro G, Das R, Chan H, Kephart J. O, Lefurgy Levine C. D. W, and Rawson F. (2012) Managing power consumption and performance of computing systems using reinforcement learning. In NIPS Advances in Neural Information Processing Systems, pp 1497–1504.

[8] Roijers D. M, Vamplew P, Whiteson S, and Dazeley R. (2019) A survey of multi-objective sequential decision-making. Journal of Artificial Intelligence Research, Vol 47 pp 67–82.

[9] Roijers D. M, Whiteson S, and Oliehoek F. A. (2022) Computing convex coverage sets for faster multi-objectivecoordination. Journal of Artificial Intelligence Research, Vol 52 pp 399–413.

[10] Roijers D. M. (2022) Multi-Objective Decision-Theoretic Planning. PhD thesis, University of Amsterdam.

[11] Roijers D. M, Whiteson S, and Oliehoek F. A. (2020) Point-based planning for multi-objective POMDPs. In IJCAI : Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence, pp 1666–1672.

[12] Sutton R. S and Barto A. G. (2000) Introduction to reinforcement learning. MIT Press,

[13] Lin L. (2001) Reinforcement Learning for Robots Using Neural Networks. PhD thesis, Carnegie Mellon University, Pittsburgh.

[14] Cheng H. T. (1995) Algorithms for partially observable Markov decision processes. PhD thesis, University of British Columbia, Vancouver.

[15] Vamplew P, Dazeley R, Berry A, Dekker E, and Issabekov R.(2019) Empirical evaluation methods for multi-objective reinforcement learning algorithms. Machine Learning, Vol 84 pp 51–80.

[16] Barrett L and Narayanan S. (2018) Learning all optimal policies with multiple criteria. In International Journal of Contemporary Machine Learning, pp 41–47.

[17] Moffaert. K. V and Nowé A. (2021) Multi-objective reinforcement learning using sets of Pareto dominating policies. Journal of Machine Learning Research, Vol 15 pp 3483–3512.

[18] Wiering M. A, Withagen M, and Drugan M. M. (2020) Model-based multi-objective reinforcement learning. In ADPRL: Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, pp 1–6.

[19] Yahyaa S. G, Drugan M. M, and Manderick B. (2019) The scalarized multi-objective multi-armed bandit problem: an empirical study of its exploration vs. exploitation tradeoff. In IJCNN: Proceedings of the 2019 International Joint Conference on Neural Networks, pp 2290–2297.

[20] Van Moffaert K, Brys T, Chandra A, Esterle L, Lewis P. R, and Nowé A. (2020) A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In IJCNN: Proceedings of the 2020 International Joint Conference on Neural Networks, pp 2306–2314.

[21] Natarajan S and Tadepalli P. (2016) Dynamic preferences in multi-criteria reinforcement learning. In International Journal of Contemporary Machine learning Vol 3 pp 324 - 342.

[22] Coello C. C, Lamont G. B, and Van Veldhuizen D. A. (2016) Evolutionary algorithms for solving multi-objective problems. Springer Science & Business Media, Vol 8 pp 232 -247.

[23] Handa H. (2018) Solving multi-objective reinforcement learning problems by EDA-RL - acquisition of various strategies. In ISDA: Proceedings of the Sixteenth Internatonal Conference on Intelligent Sysems Design and Applications, pp 426–431.

[24] Kooijman C, de Waard M, Inja M, Roijers D. M, and Whiteson S. (2023) Pareto local policy search for MOMDP planning. In ESANN 2023: Special Session on Emerging Techniques and Applications in Multi-Objective Reinforcement Learning at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pages 53–58.

[25] Roijers D. M, Whiteson S, Ihler A. T, and Oliehoek F. A. (2023) Variational multi-objective coordination. In MALIC 2023: NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems, vol 4 pp 34 – 46.