2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Accelerating Learning in Multi-Objective Systems through Transfer Learning

Adam Taylor, Ivana Dusparic, Edgar Galván-López, Siobhán Clarke, Vinny Cahill
Distributed Systems Group, School of Computer Science and Statistics
Trinity College Dublin, Ireland
{tayloral, ivana.dusparic, edgar.galvan, siobhan.clarke, vinny.cahill}@scss.tcd.ie

***Abstract*— Large-scale, multi-agent systems are too complex for optimal control strategies to be known at design time and as a result good strategies must be learned at runtime. Learning in such systems, particularly those with multiple objectives, takes a considerable amount of time because of the size of the environment and dependencies between goals.**

Transfer Learning (TL) has been shown to reduce learning time in single-agent, single-objective applications. It is the process of sharing knowledge between two learning tasks called the source and target. The source is required to have been completed prior to the target task.

This work proposes extending TL to multi-agent, multi-objective applications. To achieve this, an on-line version of TL called Parallel Transfer Learning (PTL) is presented. The issues involved in extending this algorithm to a multi-objective form are discussed. The effectiveness of this approach is evaluated in a smart grid scenario. When using PTL in this scenario learning is significantly accelerated. PTL achieves comparable performance to the base line in one third of the time.

I. Introduction

MULTI-AGENT, MULTI-OBJECTIVE LEARNING can take a significant amount of time to complete [1]. This is because the different objectives and agents affect each other making the environment more variable as well as increasing the amount of learning to do. Not only must a solution to each objective be learned, but also the agents in a system must learn how the goals interact.

Objectives' priorities can change across the state space, which means that different conditions in the environment will have different importances to different objectives. These priorities can also change over time. This makes the learning problem much more complex than in single-objective systems. Such problems can be found in many areas and are particularly common in applications with multiple stake holders such as economic systems or transportation. In these applications, each stake holder has their own particular goals –which may be mutually exclusive–, a computational entity controlling such a system needs to select which objective to obey at any one time. For example, a learning process controlling the charging of an electrical vehicle might have to find a series of actions according to two objectives: turn the device on so that it is charged for the user or leave it off to minimise the electricity drawn from the grid. These goals are contradictory and so the process will be required to resolve these conflicts for each time step. If a decision is made for each 15 minute period and the system will run for 12 hours, there will be 96 possible solutions (4 decisions per hour, 2 choices per decision). Each of these solutions is a series of actions to be taken. Some of these will prefer one goal over the other, while others will be balanced between the two goals. There almost certainly will not be a single best solution for all possible priorities, but rather the best solution will depend on how the two objectives are related. In this example, priorities could change over time, e.g. if the vehicle was fully charged after 5 hours, then the two policies would cease to be in conflict and the priorities would change from then on. This simple example only had 1 agent and two policies, and even still it would require a significant amount of learning.

In this paper, we propose Parallel Transfer Learning (PTL), an algorithm that accelerates learning in multi-objective, multi-agent systems. It does this by exploiting the similarities in what different agents learn and shares this information.

The rest of this paper is structured as follows. Section II provides a review of relevant literature. The contribution is presented in Section III, the requirements for a system implementing it are in Section IV. Issues involved in applying PTL to multi-objective systems are discussed in Section V. The idea is evaluated in a smart grid scenario in the Section VI and finally conclusions and future work are given in Section VII.

II. Background

A. Reinforcement Learning

The basic pattern underlying Reinforcement Learning (RL) [2] is to observe the environment, choose and execute an action to affect the environment and see how good that action was. An agent (a process that implements RL) can learn to perform optimally for a given state of the environment by executing this process multiple times. The agent is provided with information about how good a particular state is through a reward signal. The agent learns by adding new information to its value function. The value function is a record of how good a particular state and action has been based on the reward signal and expected future reward. When the value function converges for all states, the agent has finished learning. Learning occurs by repeatedly visiting states and seeing how actions affect the environment. Each agent must learn how its own actions affect the environment. When an agent has multiple objectives, it must learn how each action affects each objective.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

An agent's objectives are represented by a policy. A policy is a representation of all possible states of the environment and what should be done for them, they are often represented as a Markov Decision Process. These are mechanisms for describing sequential decisions in an environment. The more policies an agent has, the longer it will take for its behaviour to stabilise, as each policy requires many samples to learn from. Many RL algorithms require that each state of the environment is visited many times for the value function– and by extension behaviour– to converge.

For RL to be useful in real-world, multi-agent system scenarios (for example the smart-grid [3], [4], [5]), the learning process will need to be accelerated while exploration is occurring. The agent cannot meet its goals during exploration, as RL requires, for convergence, that every state-action pair is experienced and not all of these will be required for the optimal performance. This means that during exploration, RLs performance is necessarily suboptimal. For an agent to know a particular state-action pair is suboptimal, the agent must experience it.

Through implementing RL an agent can learn the optimal solution to a single-objective problem. When extended to multi-objective problem, further issues arise.

B. Multi-Objective Systems

Many large-scale distributed control applications require that individual entities within a system are capable of handling multiple objectives [6]. The entity will be required to find an optimal way of satisfying these objectives. This is non-trivial, as at any given time these objectives could require conflicting action, and the entity must learn a way to behave that accounts for this [7]. The problem is further compounded by the fact that the importance of a particular objective can change over time.

There are two main categories of approach to multi-objective learning in reinforcement learning:

- **Combined State Space Approaches.** These work by having a single learning process operate on a state space which is the cross-product of those of the individual objectives [8]. In this way the process learns how best to behave according to multiple objectives in the same way as with one objective. This has the added benefit of not requiring any alteration of the algorithm, just the reward scheme will change. Learning on combined state spaces can become intractable very quickly, as each additional environmental parameter or objective adds many states to the state space.
- **Arbitration-Based Methods.** In these approaches, multiple learning processes make suggestions according to their own single-objective at each time, and these suggestions are then decided between by some arbitration process [9]. There are many possible implementations for the arbiter process: election based [10], priority based [11], compromise based [9] or weight based [12].

This work will examine arbitration-based methods. This is because combined state space approaches effectively reduce multi-objective learning to a very large single-objective problem. In combined state space approaches the inter-objective priorities are encoded in the reward structure, which means that they are fixed to static values that must be known at design time. Aside from having to be known at design time, a major drawback with them is that priorities cannot change regardless of what happens in the environment. If the environment changes in a manner that would alter inter-objective priorities, the system cannot adapt without completely rebuilding the reward structure. This means they are only applicable in applications were it is known the priorities are fixed a priori.

C. Distributed W-Learning

Distributed W-Learning (DWL) [13] is a multi-agent, multi-policy, arbitration-based RL algorithm. In DWL each agent has a set of local policies. Each local policy represents a single-objective. It also has a set of remote policies, which represent the objectives of the agent's neighbours. At each time step, each policy suggests the action that it wants to execute (either to explore it or to exploit it). This produces a set of suggested actions. To decide between these actions an arbiter process looks at the W-value associated with each suggestion and selects the suggestion with the highest value to execute. The W-value [12] is a representation of how important it is that the suggesting policy be obeyed at a given time step. It is updated by the Equation 1 when the suggestion is not obeyed:

$$W_i(s) := (1-\alpha)W_i(s) + \alpha(Q_i(s,a_i) - (r_i + \gamma(\max(Q(s',a'_i))) \tag{1}$$

where $W_i(s)$ is the W-value of the state the agent is in, $Q_i(s,a_i)$ is the Q-value for the state-action that was suggested, $Q(s',a'_i)$ is the Q-value of a potential next state-action pair, r_i is the reward that would have been received, α is the learning rate, γ is the discount factor. As the value is only updated when it is not obeyed, the state has the opportunity to become more important next time if needed. This dynamically changes priorities. W-values are assigned to a state, rather than a state-action pair.

In DWL, the information about a local policy is separated away from policy relation information. This means learning each objective has two components. First the Q-component, which is the state-action values, with information on what is best for one objective. It contains no information about how actions affect other objectives (or those of other agents). The second part is the W-component, which contains only information on how actions affect other policies, either local or remote, in the system.

D. Accelerating Learning

There have been many attempts to accelerate learning in single-objective, reinforcement learning systems. The way the value function is represented can be changed. For example, a function can be used instead of a table [14]. Using a good function can allow the state space to be compressed and

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

similar states to be grouped. This means that knowledge can be generalised and thereby more efficiently used. However, if the function is poorly chosen, it can lead to inaccurate learning or reduced accuracy of representation.

The algorithm used to explore the state space can be altered or changed. For example, Prioritised Sweeping [15] is a modification to standard RL algorithms. It maintains a list of states that have led to the current state, and when a reward is received it is then propagated back accordingly. This means that the values are more quickly propagated back to the correct states and less learning time is required.

Learning can also be accelerated by the addition of external knowledge. Several different approaches as to how knowledge can be incorporated have been explored [16]. The authors find that adding external knowledge through action selection is preferable as it has no lasting effect on the value function. Heuristically accelerated Q-Learning [17] is one such approach. It works by defining a function which biases action selection, which encourages exploration in areas thought to be particularly important or beneficial. The knowledge encapsulated in the heuristic function can come from any source, for example prior experiences or human knowledge.

Genetic algorithms have also been applied to multi-objective learning and have been shown to be effective [18]. They can also be combined with RL [19]. The exploration of the state space is done using policy search with a set of potential solutions, and these policies are then evaluated and modified. Over time, the correct solution will emerge. This can only improve the speed of learning if the problem can be run several times simultaneously.

E. Transfer Learning

Transfer Learning (TL) for RL [20] is an emerging field in this area. It is based on an idea borrowed from psychology. When learning how to accomplish a task, knowledge from a related task is often used as a starting point. When applying this concept to RL, it can be accomplished by sharing information about states of the environment and good actions amongst different agents. This is accomplished by completing learning in one task (called the source task), mapping the information from source to target (effectively translating it, so it is intelligible and applicable to the target) and passing it to the target agent, which then incorporates this new knowledge into its state space. This is all done off-line, prior to the execution of the target task.

A mapping is used to allow a common understanding of what data means. This is typically achieved through a function, which takes knowledge from the source and translates it so it is in the same form as knowledge the target produces itself. In the literature, this mapping has typically been hand-coded by the designers [21], which is not a scalable solution. In work by Ammar et al. [22], a mapping function was derived automatically. This type of autonomous mapping will be required for real-world applications of TL.

The goal of TL, as indicated before, is to reduce learning time in the target task. Providing information from a good source has been shown to significantly accelerate learning in the target [23] but a large amount of time is required to learn in the source task which is not reduced. If this time is considered as part of the execution time for the target, then the increase in speed is significantly reduced [22].

In this paper this type of TL where learning in the source task happens prior to the learning in the target task, will be called Sequential Transfer Learning (STL). The transfer is unidirectional following a sequential order (see Figure 1a). This figure shows the flow of data in sequential TL. Source Task 1 is passing information to Target Tasks 1 and 2, while Source Task 2 provides data to Target Tasks 2 and 3.

If these agents have multiple policies, then information about how policies interact can also be shared. Sequential TL will be difficult to apply to multi-policy agents, as the relationship between policies on one agent is dynamic and complicated [24]. Even if both source and target tasks learned in identical scenarios, the policy dependencies may differ [25]. This is particularity true if there are several Pareto-optimal[1] solutions. This means that a single transfer of policy dependence information at the start of the learning process will not adequately represent relationships in the target task even if they are closely related as it changes over time. The process of learning in the target agent will invalidate the information received from the source. For this reason a single transfer will not be particularly useful when the target task that is required to learn has a dissimilar set of policies or operates in a different environment. As many real-world systems are multi-objective, extending TL to work in theses would be desirable and is discussed below.

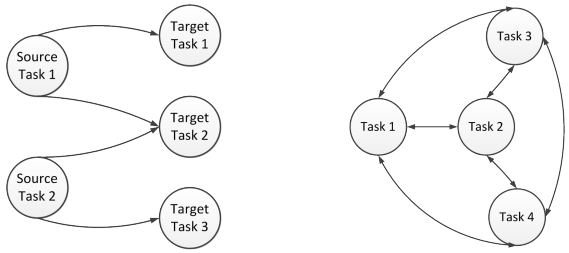

Fig. 1: Information flow in (a) Sequential Transfer Learning (b) Parallel Transfer Learning.

III. Parallel Transfer Learning

Parallel Transfer Learning (PTL) [27] is the idea of having both source and target task learn simultaneously. The source shares information when it considers the information to be relevant to the target task. This removes the requirement for having learned source information prior to executing the target task and it also allows for multiple transfers. Multiple transfers enable policy dependence information to be consistently supplied, which means that the dynamicity of multi-objective learning can be transferred to the target

[1]Pareto optimality is a balance between objectives in which improving performance on one objective necessarily makes another perform worse [26].

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

agent. Whenever the relationship between policies changes, relevant information can be communicated. The disadvantage with this approach is there is no learned information available at the start of learning in the target task, so there is initially no benefit in terms of performance. When using STL, the target task begins learning with at least some knowledge, this means that it can perform comparatively well initially. It takes some time for PTL to overcome this initial lack of knowledge, and the amount of time is application dependent. There is no reason the two could not be used in conjunction to overcome this if information were initially available. The roles of source and target in PLT are more fluid. Any particular agent can act as a source, target or both at a given time, as the transfers can go between any pair of agents in the system. This is shown in Figure 1b, where at any one time transfers could be occurring through any set of the arrows. Different approaches to selecting data to transfer and how to receive it are discussed in [27].

IV. Design

A system implementing PTL is composed of the following elements:

- A set of agents, $A = \{A_1, ..., A_Z\}$, where Z is the total number of agents in the system.
- A set of neighbours for every agent, $N_i = \{N_{i,1}, ..., N_{i,Y}\}$ where Y is the number of agents the agent A_i is transferring to or from.
- A set of local policies for every agent in the system, $LP_i = \{LP_{i,1}, ..., LP_{iX}\}$, where X is the number of local policies implemented by the agent A_i.
- A set of mappings $\chi_{i,j}$ from each of agent A_i's local policies LP_i to each the policies belonging to the agents in its neighbour set N_i. This is given by $\chi_{i,j} = \{\chi_{i,j\to1,1}, ..., \chi_{i,j\to Y,X}\}$, where $\chi_{i,j\to1,1}$ is the mapping from the agent A_i's jth policy to the agent A_1's first policy, where A_1 is the first agent in N_i, the neighbour set of the agent A_i.
- A set of transfers $\tau_t = \{\tau_{i,t}, ..., \tau_{Z,t}\}$ where $\tau_{i,t}$ is the set of transfers being received by the agent A_i at time step t.
- Each mapping and transfer will have two components, a Q-component and W-component given respectively as the following: $\chi_{i,j} = \chi^Q_{i,j} + \chi^W_{i,j}$, $\tau_t = \tau^Q_t + \tau^W_t$

At each time step an agent A_i will select a subset N_{itrans} of its neighbours N_i. A_i then selects data items from a subset LP_{itrans} of its local policies LP_i. Each of these data items is then put through the mapping in the set $\chi_{i,j}$ corresponding to its intended recipient. This produces A_i's component of the set of all transfers τ_t. This set is then sent over the network. Upon receiving transfers to it $\tau_{i,t}$, A_i merges them into its state space.

V. PTL for Multi-Objective Systems

There are two types of knowledge which can be transferred. The first is information relevant to one objective. In RL terminology, this might be the value associated with a particular state-action pair. The other option is to transfer objective-relation information, which could take the form of policy weightings. The usefulness of both data types will change over time. As a result, the way data is selected to be transferred and how it is received will also change.

A. Q-Transfers

When agents are learning and exploring in a multi-objective system, most schemes require them to learn how to first satisfy each objective individually. During these single-policy explorations, more focus on Q-component transfers will most benefit learning, as at this time the agent is sacrificing attempting to satisfy multiple goals, and is instead trying to gain knowledge.

When deciding what Q-component information should be shared at any one time, there are two main aims:

- To share information learned locally that is representative of the final converged value.
- To ensure that information received via transfer is propagated through the system. Not all agents will have the same sets of neighbours, so multiple agents may be required to fully disseminate a piece of transferred information.

Linked to this is the decision of how frequently to share data and when to do so. At one extreme data could be shared every time it changes, at the other it could be shared only when it stabilises to a particular value (that being an approximation of when learning is finished). Sharing data too frequently or too early can lead to it being misleading as one experience is not necessarily representative of subsequent ones. This is particularly true in dynamic environments or those with multiple agents. If information is shared too infrequently, then it may not be available to the target agent when needed. This opportunity cost can be particularly pronounced in seldom visited states. In these states, a single experience by one agent could be valuable to the system overall if properly shared. It will take several visits from each agent to converge that state, but given the state is seldom visited, there will be time to completely share the experience, so in aggregate the system will need to visit the state far less to learn how best to act.

There are several schemes to select data that meet these requirements. Which one works best will be application dependent as different environments will have differing degrees of change. Transferring states that have been visited most will tend to share the values that there is the most confidence in, however it will prefer the most common states and will share only a small set of states (those that occur frequently) neglecting less common ones. Selecting the state that has shown the greatest degree of change over a period of time will work well for sharing recently visited states and pass on received information but will tend not to share near-converged values, because close to convergence there is only very small changes in value.

When merging information, an agent needs to consider what it knows already. If it has no experience of a particular

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

state, then it can just accept what it is being told completely (this ignores the possibility of malicious agents). The more interesting case is when it already has information about the state in question. In this case, the agent must decide if the new information is 'better' that its own. It would be possible to share meta-information stating how many times the state was visited but this will be affected by different learning rates and cannot be easily mapped between heterogeneous agents. Agents, therefore, will have to judge how to merge a transfer based solely on its contents. They can use their own value for a state as a heuristic for what the state's final value might be then if the received information is consistent with this it can be accepted. Alternately, the two sources of information could be added as a linear combination. This allows the weight given to received information to decay overtime, so as an agent has more local information it cares less about the information others provide.

B. W-Transfers

Transferring W-components has similar issues to changing Q-components but additionally interdependencies must be accounted for. When changing the value of a particular state-action pair, the only effect it has is how frequently that action is selected at that state for that objective. Changing a W-component can affect the balance that has been learned. Multi-objective learning effectively finds a balance between the objectives and changing even one W-value can result in very different behaviour. This means that W-component transfer should only occur during the exploration phase.

When there are policies that require collaboration (such as those concerning a shared resource), it will be necessary to maintain some degree of W-value diversity. Effectively agents learn a way of interleaving or balancing their use of the shared resource. If W-values are over-transferred this balance may be affected. For example, if three agents are trying to charge an EV, each from a limited electricity supply, one solution they may learn is effectively round-robin. In this, each one would charge every third time step, with an offset to prevent collision. If one of these agents was assigned the same W-values as one of the others, they would try to charge at the same time. This would cause uneven loading on the transformer, the very thing they had learned to avoid. For this reason, only a small subset of all W-values should ever be transferred. Thus, the exact number is application dependant.

VI. Experimental Scenario

The evaluation of PTL is done using a multi-objective, smart grid scenario. The smart grid [28] involves applying computational intelligence to the electrical grid. The evaluation focuses on a demand response [29] scenario. Demand response is the idea of changing electrical demand to react to supply or other factors (such as price, low-carbon energy etc.). For example, if on a windy day, there was a surplus of wind energy generated, electrical devices implementing demand response could switch on and draw more energy. This would mean they then would not need to operate at another time. If demand can be intelligently modulated in this way, the electrical grid can operate much more efficiently [30]. The degree of flexibility that devices can exhibit when implementing demand response depends on their function. An Electric Vehicle (EV), for example, can shift when it charges significantly as long as it is sufficiently charged when it is to be used. Electric lighting cannot be rescheduled, as it must operate when turned on. This evaluation will focus on EVs, as their flexibility makes them interesting from a multi-objective point of view. The more a device can be rescheduled, the more Pareto-optimal solutions there are.

Method	PAR	Ave. Charge Range (%)	Ave. Delta (W)
Probabilistic	1.95	66-100	837
DWL	1.69	66-100	3011
DWL+PTL(Q+W)	1.68	64-98	1278

TABLE I: The Results of Method Comparison.

The simulations were conducted using GridLAB-D [31], which is an open-source, electrical grid simulator developed by the US Department of Energy. The scenario simulated has nine EVs, which implement demand response through a DWL agent. Each agent has three policies:

- A user policy - The EV must be sufficiently charged when required by the user.
- A transformer policy - The EV should try to minimise the load placed on the transformer at any one time.
- A predicted load policy - The EV should try to minimise the number of times the transformer is overloaded and smooth loading over time.

Each EV requires approximately 7 hours of charging to meet the requirements of their daily journey. The base electricity consumption is taken from an Irish Smart Meter trial [32]. The results for the transformer policy will be given in terms of peak-to-average-ratio (PAR) [33]. As a performance metric, PAR is clearer than raw transformer load profiles. The predicted load policy can be judged based on the average δ. This is calculated by averaging the magnitude of the change in transformer load between time steps. When applied to DWL this will be calculated for the exploitation phase only. This is a metric representing the 'smoothness' of the load profile. Jagged loading is undesirable for the physical transformer as it can cause wear. The user policy's main metric is the average battery charge range during exploitation. This is the average starting charge and average final charge. The user policy is significantly easier to satisfy that the others. It is affected only by the agent's own actions; if the agent chooses to charge, then the battery gets more charge. The other policies depend on what other agents in the environment do. When trying to limit the peaks, the agent can be penalised based on others' actions. All experiments were run 10 times and averaged.

A. Method Comparison

Table I shows how the DWL+PTL(Q+W) (transferring both Q and W-components) scheme described above works

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

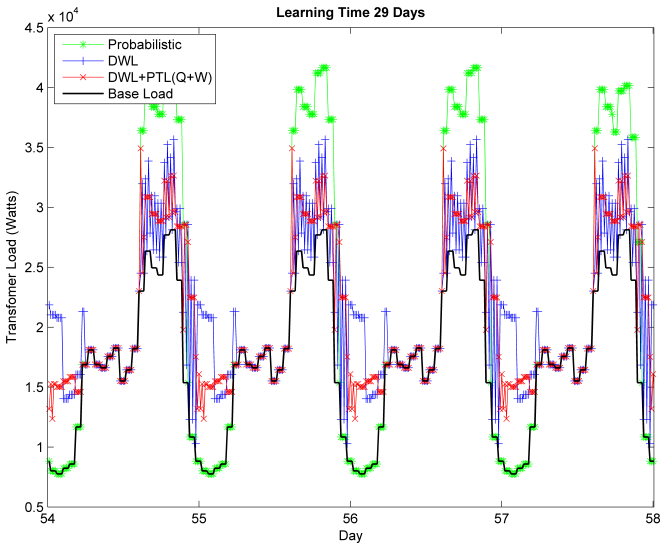

Fig. 2: The transformer load of different schemes over four days.

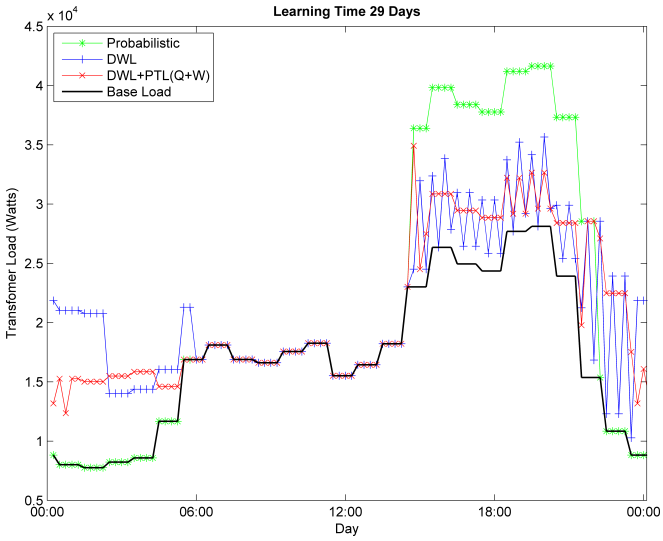

Fig. 3: The transformer load of different schemes over 24 hours.

when compared to an approach implemented in GridLAB-D, which is effectively a greedy solution to the user policy with some probabilistic load balancing (called probabilistic in tables and figures). Both DWL schemes had 29 days of exploration, before exploiting their learned solution. The Figure 2 shows the load on the transformer over a four day period, Figure 3 shows one day's load. The pattern in these figures of a medium load in the mornings, an evening peak and a trough over night is due to both the base load (the solid black line on the graph) and the unavailability of EVs to charge for a period around 02:00 due to them having finished charging. The line for the probabilistic approach (green with * markers) has much higher peaks each evening as it charges EVs probabilistically as soon as they are available. This means when the base load is low there is no extra load to add

Training	PAR	Ave. Charge Range(%)	Ave. Delta (W)
9 days	1.95	59-92	806
19 days	1.84	66-100	775
29 days	1.8	66-100	1131

TABLE II: The Results of DWL over time.

Training	PAR	Ave. Charge Range(%)	Ave. Delta (W)
9 days	1.83	66-100	2898
19 days	1.68	57-90	1280
29 days	1.69	61-94	1285

TABLE III: The Results of DWL+PTL(Q) over time.

(seen where the Probabilistic line and base load are equal). In contrast the two DWL based approaches shift some of the load from the evening peak to the overnight trough. This is the essence of demand response.

An evolutionary based approach to the same scenario is used in [34], where an average PAR of 2.24 is obtained. All approaches tested here and the evolutionary approach are capable of meeting the user's demands. The PAR shows that both DWL-based methods perform best according to the transformer policy. The average δs show that there is similar smoothing performance (the goal of the predicted load policy). One individual charger draws 1430 Watts, so the probabilistic approach and DWL+PTL(Q+W) are averaging less than one change in the number of charging EVs per time step. Minimising the number of times a charger is turned on and off improves the length of its operational life.

In this set of experiments, DWL+PTL performs best on average over all objectives. The following experiments will investigate how different combinations of transfer affect results.

B. Learning Time Experiments

The experiments discussed in this section illustrate how PTL affects learning time, they use the previous scenario, only exploration time changes. They are judged based on performance, thus if the system reaches a performance of P in time T using only DWL and with PTL it can reach P in $T/2$, then PTL has learned significantly faster. The experiments will be run with three different learning periods: 9, 19 and 29 days. Four schemes will be compared, DWL, DWL+PTL(Q), DWL+PTL(W) and DWL+PTL(Q+W).

The Tables II - V show the four schemes' results. Each of the different schemes tends to find a slightly different balance between policies. This is particularly true after limited training (see 9 day training results). In these, there has not been enough time to find a balanced solution, so when

Training	PAR	Ave. Charge Range(%)	Ave. Delta (W)
9 days	1.79	59-92	2494
19 days	1.72	66-100	1247
29 days	1.69	61-94	1445

TABLE IV: The Results of DWL+PTL(W) over time.

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

Training	PAR	Ave. Charge Range(%)	Ave. Delta (W)
9 days	1.65	61-94	1404
19 days	1.62	66-100	1382
29 days	1.68	60-94	1247

TABLE V: The Results of DWL+PTL(Q+W) over time.

the agents exploit knowledge, they tend to find a solution that is good for one policy, as these are easiest to learn.

The DWL scheme initially prioritises the predicted load policy, which leads to good smoothing but rather poor performance on the other metrics. Similarly, the DWL+PTL(W) scheme reaches a good PAR at the expense of smoothing performance. The performance of only DWL after 29 days of exploration is comparable to that of the PTL schemes after 9 days (approximately a third the learning time). It takes PTL far less time to get the majority of exploration done. PTL finds a reasonably effective solution with very little exploration, the honing of this solution to one which performs well on all three policies continues after this. This accounts for the relatively minor fluctuations in performance through 19 and 29 days of exploration. The trade-off between the three policies becomes apparent once the PAR reaches about 1.7. To further reduce the PAR, the agent must charge less during the evening when the base load is high (and contributing to the total peak). If it does not charge during the evening, it leaves a lot to do during the rest of the night and morning. This risks an EV not being charged when required, this results in a very large negative reward, which in turn makes the user policy's priority increase when there is a chance it won't be met. This effect can be seen in the Table III, where the PAR actually increases with more training. In this case it has accepted worse performance on the transformer policy in favour of the user policy.

VII. Conclusion and Future Work

This paper proposes an algorithm to accelerate learning in an arbitration base approach to multi-objective RL, Distributed W-Learning (DWL). This algorithm allows the relatedness of tasks in a multi-agent system to be exploited to accelerate learning. This is achieved by reusing information learned by one agent in another's learning process. Each agent passes information to other agent, which allows them to learn from each other's experiences. The algorithm is called Parallel Transfer Learning (PTL).

The results presented here show that learning time can be improved by using PTL. It is particularly effective early in the learning process when there is little information available. In this paper, it was evaluated in a multi-objective, smart grid problem. In this scenario, DWL+PTL achieves comparable performance to DWL with a third of the learning time.

It is planned to extend this work by introducing inter-policy mapping, which is the idea of taking information from one policy and applying it to a distinctly different one. It requires that the information is 'translated' so the target policy can understand it. Potentially this could significantly improve the speed of learning, as learning what is best for one policy would be equivalent to learning what is best for all. It is unlikely to be quite that effective in most scenarios, it might be able to speed up initial learning significantly. This type of mapping would need to be produced by the agents themselves so that transfer could occur between arbitrary policies deployed in a real system.

We would also like to investigate how PTL could be tweaked to make it more effective. This might include developing a heuristic as to which pairs of agents might have useful information to share and which states are particularly important to which policies. This heuristic could then be used to guide the selection of data to transfer. Using a PTL scheme it will be possible for agents to cooperatively explore the state space, this has the potential to significantly accelerate learning, as the state space's size will be effectively reduced to a fraction of its original size, where the fraction is one divided by the number of cooperative agents.

Acknowledgement

This work was supported by Science Foundation Ireland under the Principal Investigator research program 10/IN.1/I2980 "Self-organizing Architectures for Autonomic Management of Smart Cities".

References

[1] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[2] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, 1999.

[3] A. Dimeas and N. Hatziargyriou, "Multi-agent reinforcement learning for microgrids," in *Power and Energy Society General Meeting, 2010 IEEE*. IEEE, 2010, pp. 1–8.

[4] E. Galvan, C. Harris, I. Dusparic, S. Clarke, and V. Cahill, "Reducing electricity costs in a dynamic pricing environment," in *Proc. Third IEEE International Conference on Smart Grid Communications (SmartGridComm)*. Tainan, Taiwan: IEEE Press, november 2012, pp. 169 – 174.

[5] E. Galván-López, C. Harris, L. Trujillo, K. Rodriguez-Vazquez, S. Clarke, and V. Cahill, "Autonomous demand-side management system based on monte carlo tree search," *International Energy Conference (EnergyCon), Dubrovnik, Croatia*, 2014.

[6] I. Dusparic and V. Cahill, "Multi-policy optimization in self-organizing systems," *Self-Organizing Architectures*, pp. 101–126, 2010.

[7] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[8] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Hypervolume-based multi-objective reinforcement learning," in *Evolutionary Multi-Criterion Optimization*. Springer, 2013, pp. 352–366.

[9] I. Dusparic and V. Cahill, "Autonomic multi-policy optimization in pervasive systems: Overview and evaluation," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 1, p. 11, 2012.

[10] J. D. Siirola, S. Hauan, and A. W. Westerberg, "Computing pareto fronts using distributed agents," *Computers & chemical engineering*, vol. 29, no. 1, pp. 113–126, 2004.

[11] D. Houli, L. Zhiheng, and Z. Yi, "Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, p. 7, 2010.

[12] M. Humphrys, "Action selection methods using reinforcement learning," *From Animals to Animats*, vol. 4, pp. 135–144, 1996.

[13] I. Dusparic and V. Cahill, "Distributed w-learning: Multi-policy optimization in self-organizing systems," in *Self-Adaptive and Self-Organizing Systems, 2009. SASO'09. Third IEEE International Conference on*. IEEE, 2009, pp. 20–29.

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.

[14] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.
[15] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, no. 1, pp. 103–130, 1993.
[16] W. B. Knox and P. Stone, "Reinforcement Learning from Simultaneous Human and MDP Reward," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, 2012, pp. 475–482.
[17] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa, "Heuristically Accelerated Q Learning : A New Approach to Speed Up Reinforcement Learning," in *XVII Brazilian Symposium on Artificial Intelligence - SBIA'04*, 2004, pp. 245–254.
[18] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," *Lecture notes in computer science*, vol. 1917, pp. 849–858, 2000.
[19] J. J. Grefenstette, D. E. Moriarty, and A. C. Schultz, "Evolutionary algorithms for reinforcement learning," *arXiv preprint arXiv:1106.0221*, 2011.
[20] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains : A Survey," *The Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.
[21] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions onransactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.
[22] H. B. Ammar, K. Tuyls, M. E. Taylor, K. Driessens, and G. Weiss, "Reinforcement Learning Transfer via Sparse Coding," in *AAMAS 2012: Proceedings of the eleventh international conference on autonomous agents and multiagent systems*, no. Aamas, 2012, pp. 4–8.
[23] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *Journal of Machine Learning Research*, vol. 8, pp. 2125–2167, 2007.
[24] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2011.
[25] A. Castelletti, F. Pianosi, and M. Restelli, "A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run," *Water Resources Research*, 2013.
[26] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and computers in simulation*, vol. 60, no. 3, pp. 245–276, 2002.
[27] A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill, "Transfer Learning in Multi-Agent Systems Through Parallel Transfer," in *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning (Poster)*, vol. 28, 2013.
[28] S. Massoud Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005.
[29] R. Fazal, J. Solanki, and S. K. Solanki, "Demand response using multi-agent system," *2012 North American Power Symposium (NAPS)*, pp. 1–6, Sep. 2012.
[30] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in smart grid using heuristic optimization," *Smart Grid, IEEE Transactions on*, vol. 3, no. 3, pp. 1244–1252, 2012.
[31] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *Transmission and Distribution Conference and Exposition, 2008. T&D. IEEE/PES*. IEEE, 2008, pp. 1–5.
[32] Comission for Energy Regulation, Ireland. (2011) Smart meter trial data. [Online]. Available: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/
[33] A. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *Smart Grid, IEEE Transactions on*, vol. 1, no. 3, pp. 320–331, 2010.
[34] E. Galván-López, A. Taylor, S. Clarke, and V. Cahill, "Design of an automatic demand-side management system based on evolutionary algorithms," *Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, Gyeongju, Korea*, 2014.

Authorized licensed use limited to: Maynooth University Library. Downloaded on January 31,2022 at 14:12:58 UTC from IEEE Xplore. Restrictions apply.