Transfer Learning in Multi-Agent Systems Through Parallel Transfer

Adam Taylor TAYLORAL@TCD.IE
Ivana Dusparic IVANA.DUSPARIC@SCSS.TCD.IE
Edgar Galván-López EDGAR.GALVAN@SCSS.TCD.IE
Siobhán Clarke SIOBHAN.CLARKE@SCSS.TCD.IE
Vinny Cahill VINNY.CAHILL@SCSS.TCD.IE
School of Computer Science and Statistics, Trinity College Dublin, College Green, Dublin 2, Ireland

Abstract

Transfer Learning(TL) has been shown to significantly accelerate the convergence of a reinforcement learning process. TL is the process of reusing learned information across tasks. Information is shared between a source and a target task. Previous work has required that the target task wait until the source task has finished learning before transferring information. The execution of the source task prior to the target task considerably extends the time required for the target task to complete. This paper proposes a novel approach allowing both source and target task to learn in parallel. This allows the transfer to be bi-directional, so processes can act as both source and target simultaneously. This, in consequence, allows tasks to learn from each other's experiences and thereby reduces the learning time required. The proposed approach is evaluated on a multi-agent smart-grid scenario.

1. Introduction

The basic pattern underlying Reinforcement Learning (RL) (Sutton & Barto, 1999) 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. In large-scale, multi-agent systems, to learn to achieve acceptable performance can take a significant amount of time (Busoniu et al., 2008). Each agent must learn how its own actions affect the environment. In the single-agent case, this is achievable as the agent can try an action and observe the result. In the multi-agent case, this approach becomes significantly slower and less reliable as the environment is also being affected by the actions of neighbouring agents. The actions taken by other agents in the system can cause the outcome of an agent's action to differ from previous experiences. If an agent selects a constant action for a given state of the environment, its results will vary based on what other local agents (neighbours) do as each neighbour is also affecting the environment. As the number of agents grows, this effect becomes more extreme.

An agent has goals that it is trying to learn to satisfy; a goal is represented by a policy. Learning in multi-agent systems is made more complex when multiple policies are considered. Each policy can, at any time, be used to select an action. This increase in the variability in action selection for a particular agent will cause the agents behaviour to be more dynamic than in the single-policy case. As RL requires that each state of the environment be visited many times so that the system can learn to perform well in that particular state, the larger the environment the longer it will take for the value function to converge.

For RL to be useful in real world multi-agent systems such as the smart-grid (Dimeas & Hatziargyriou, 2010; Galvan et al., 2012), the learning process will need to be accelerated while exploration is occurring, the agent cannot meet its goals. Some substandard performance is necessary, as for an agent to know a particular action is suboptimal the agent must experience it. There have been many attempts to accelerate learning in RL (Powell & Ma, 2011; Drummond, 2011). Transfer Learning (TL) for RL (Taylor & Stone, 2009) is

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

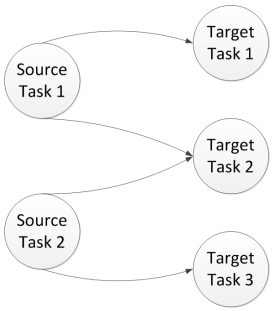

Figure 1. Information flow in Sequential Transfer Learning.

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. The agent that provides the experience to others is called the source task, those that receive information are called target tasks. 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 (Taylor et al., 2007) 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 (Ammar et al., 2012). In this paper this type of TL will be called sequential because, the source task is executed prior to the target task. The transfer is unidirectional following a sequential order (see Figure 1). 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. This all happens prior to the execution of the target tasks. In this work evaluation will be done in a smart-grid scenario. Agent-based approaches are of particular applicability to the smart-grid. The smart-grid vision (Massoud Amin & Wollenberg, 2005) applies information and communication technologies to the current grid while incorporating distributed renewable generation. Due to the wide scale of the electrical grid, centralised control is impractical. A distributed solution will need to be self-organising to allow for changes in structure, usage and demand. This solution will need to be capable of adapting to the changing environment and improving its performance over time. This will allow for the continued adaptation of distributed, renewable, electricity generation.

The rest of this paper is organised as follows: the next section will provide background in reinforcement learning and transfer learning. Following that, in section 3 this paper's contribution, parallel transfer learning, will be discussed. The experiments conducted using parallel transfer learning will be discussed in section 4. Finally, this paper closes with conclusions.

2. Background Information

2.1. Reinforcement Learning for Multi-Agent Systems

A Reinforcement Learning (RL) (Sutton & Barto, 1999) system has an agent and a representation of the environment. The agent will have a reward function, a policy and a value function. A policy is effectively a set of instructions telling the agent what to do in a particular circumstance. The reward function informs the agent how good a particular state of the environment is (there is not necessarily feedback for every state). The value function represents how good the current state of the environment is, it is usually based on expected future reward as well as current reward. The value function has a value for each state, as the agent learns the value function moves towards the true value for each state[1]. The goal of RL is to find the optimal policy. The optimal policy is the sequence of actions that return the highest cumulative reward value possible. As the reward value represents the agent's goals, the optimal policy is the best way to satisfy them. RL requires no user input to function, as a result it is applicable for use in large-scale, self-organising systems.

RL often uses a Markov Decision Process (MDP) to describe the problem space and environment. In an MDP, one combination of the parameters that describe the environment is called a state, the MDP contains all possible states of the environment. The complete set of all states is commonly called the state-space. Each state has a set of actions that are available to the agent at that state. The actions cause the environment to transition to another state (or remain in the same state).

Multi-agent systems (MAS) (Sycara, 1998) are systems consisting of more that one intelligent agent. MAS can be either collaborative, competitive or the agents can be unaware of each other (Stone & Veloso, 2000). In MAS the global state of the system is a combination of all local states; a global action is the

[1]RL agents will only converge to a fixed value if the reward function and the environment stabilises. As this is unlikely in a multi-agent system, the term 'true value' will be used to mean a value that the agent appears to settle at, there is no guarantee this value is final.

combination of all local actions. This leads to an exponential increase in the state-action space (Guestrin et al., 2002). The size of the state-space in MAS is normally enormous making it intractable for the agents to learn effectively. There are two approaches to learning in MAS. Agents can optimise their own actions in a hope to reach the global optimum these are called independent learners. Joint action learners collaborate with a subset of agents in the system to find locally and globally good decisions (in the scale of the agents involved in the collaboration). The idea is that from combining these partial global actions, a good global action will emerge (Nikos, 2003).

Joint action learners (JAL) need to communicate or observe the actions of their neighbours (Bianchi & Bazzan, 2012). JALs need a representation of how their own actions affect the environment and how their actions affect their neighbours. The representation used is usually an MDP for the agent's own actions and view of the environment and others for each of its neighbours.

2.2. Transfer Learning in Multi-Agent Systems

At a high level, there is only one requirement to successfully transfer information from one task to another. That is a common understanding between both tasks that allows information from one task to intelligible to the other. Commonly, this is achieved through either a shared representation (i.e., both problems have been cast so they have the same representation) or some form of translation is done. If the agents are homogeneous, they already share a representation, so no translation is required. Alternately, when transferring between heterogeneous agents, the information will need to be recast so that the target task can understand it. This is necessary as different agents will have different ways of representing the environment related to their individual concerns.

Determining how information should be translated is an open problem in TL (Pan & Yang, 2010). In past work, the mapping of information from source to target has been hand-coded (Taylor et al., 2007). This is obviously not a scalable solution and if TL is to be applicable to self-organising systems there needs to be a way for a system to determine how to map information independently. Ammar & Taylor (2012) attempt this through the use of a common subspace. The subspace is of a lower dimensionality than that of either the source or target; it is composed of shared features found in both tasks. Each state in source and target is then projected on to the subspace. Each state in the source is matched to a state in the target. In a more recent work by Ammar et al. (2012), an approach is adopted that maps tasks onto a high dimension intermediate space. This approach is broadly similar to the above work in that it uses a common subspace to facilitate mapping. The major difference is the dimensionality of the common subspace when compared to that of the source and target tasks and the process that produces it. The dimensionality of the intermediate space can be used to provide a heuristic as to if there is useful knowledge to transfer. If the common space's dimensionality is very high (close to the source's plus the target's dimensionality) when compared to that of the other tasks, it is likely worse as little commonality between source and target has been found. This follows as two completely unrelated tasks will often have very little overlap in the basis vectors needed to represent each of them individually.

Most work on TL for RL has focused on single-agent systems. The MAS could benefit from transfer learning as there are several different scopes for transfer to occur at. It can occur within one agent, transferring from an MDP containing information about how actions affect one neighbour to another for a different neighbour. It can also happen between agents. In this case the agent could transfer information about how its own actions affect it or how its actions affect its neighbours. Transfers outside of these would be possible but are less likely to have useful information to share. The problem of determining if there is useful information to share is discussed below.

Boutsioukis et al. (2012) focuses on evaluating the applicability of transfer to multi-agent RL. They discuss transferring from a single-agent system to multi-agent one and from one MAS to another. They particularly focus on the issues in applying transfer to MASs. Many of the problems are similar to those in the single-agent setting. They propose a framework in which mapping from source to target can be achieved by generalising knowledge dependent on the number of agents or information specific to an individual agent. In their experiments they find that transfer can be effective in MASs but that using a multi-agent source task has no benefit over a single-agent source task. As these are empirical results there is no guarantee they will generalise, particularly given the complexity of MAS.

Another limitation of TL is that there is no guarantee that there will be any benefit to doing transferring information. While transfer learning can improve performance, it can also be detrimental to it, this is called negative transfer (Taylor & Stone, 2009). To avoid this, the tasks chosen should be closely related,

this increases the likelihood that there is useful information to share. For example, an agent that solves a maze could be the source task for one which must learn to avoid obstacles. In some instances tasks will only be partially related and in these cases only some knowledge should be transferred. There is no reliable way of determining prior to performing transfer learning if there is useful information to transfer. This issue is related to selecting what to transfer and when in Parallel TL and will be examined later in this section.

Taylor & Stone (2009) propose a set of metrics than can be used to determine if TL is proving effective. The metrics cover three of the possible benefits of TL; better early performance, better final performance and speed of learning. All three are required as TL effectively changes the way in which an agent explores its environment. The knowledge provided changes the policy the agent learns, it may prove better on some of the metrics but not others.

The calculation of the time required for a TL agent's learning to stabilise (this does not mean it has finished learning or even that the value function is correct, it just means that the actions selected as best for a particular state are no longer changing with time) can be calculated in two ways. It can be the time from when the target agent begins learning having already been given the source's information or it can include the source task time and mapping time. The former is only applicable when the source task has already been executed and its information can effectively be recycled. If the source task must be executed specifically to speed up the target task then its time should be counted. Both are applicable as in a real world system both scenarios will arise. For example in a MAS it is possible that a library of possible source tasks could be maintained and looked up whenever a new learning process was to start, this would use the latter method.

In summary, prior work on TL for RL has been focused mainly on single-agent systems. This meant that learning in the source task had to occur prior to learning in the target task. This sequential approach could also be applied to MAS. However, doing so has several drawbacks. It does not take advantage of the fact that agents in the MAS are related to each other and are likely learning similar things. It would require source tasks to be selected for each individual MAS that will have useful information to transfer. This introduces a considerable amount of preparatory work to deploying a system. For use in self-organising systems, the process of selecting suitable source task may not be possible to automate. Finally the source information must be available before an agent begins learning; the source tasks have to be run. This will further extend the time required to deploy such systems.

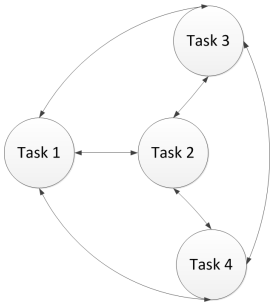

Figure 2. Information flow in Parallel Transfer Learning.

3. Parallel Transfer Learning

To address the issues of requiring source information and selecting source tasks, this work proposes Parallel Transfer Learning (PTL). In PTL the source and the target task can run simultaneously (see Figure 2, for example). This means at the beginning of a PTL system there is no learned information available to be transferred. Instead, transfer occurs when it is deemed to be useful to the target agent. By having source and target learn at the same time, there is no need for the transfer process to be unidirectional. Any pair of agents may have had different experiences, this means that both will have information useful to the other. They need not begin learning at the same time, this gives much more flexibility to learning in MAS as not all agents need to be present at the start of the system (i.e., an agent joining the system can benefit from the experiences of agents already present). Learning can overlap, happen concurrently or sequentially. The Figure 2 shows potential transfers between pairs of agents. These transfers could take place at any time during the execution of the MAS.

At each time step, if an agent decides it has useful information to share with another agent, it sends the information to that agent. It then checks its own communications buffer to see if any information has been received from other agents in the system. If it has, it decides whether to accept this information or discard it. Through repeating this procedure, over time learned information will propagate through the system.

Parallelising TL in this way removes the need for running the source task prior to the target; this overcomes some of the issues with sequential TL. As the agents in a MAS will likely be learning similar value functions, it removes the need to identify good source tasks. It

can efficiently utilise rare experiences. If one agent experiences a state that is seldom visited, this experience is essentially more valuable as this particular state will take a considerable amount of time to converge. By being able to share this with its peers, performance in these rare states can improve much faster than it otherwise would. This approach is particularly advantageous if an agent joins the system after it has run for some time. In this case there will already be good information available, so the joining agent's initial performance will be improved significantly by the knowledge of its peers.

An agent acting as both a source and target is compatible with the idea of sequential TL. There is no reason a single target task cannot have several source tasks. This means that an agent in a system need not be limited to only transferring to and from one other. It can select partners based on the information they have and how applicable it is. This means the whole MAS becomes more of a learning mesh than a collection of learning agents. Through the interchange of transfer partners useful information will be distributed much more quickly that if it were being learned by sequential TL learners.

There are several interesting questions to be answered in the design of a PTL system.

- What particular states have interesting information to transfer?
- When should this information be shared and how frequently?
- How should received information be incorporated into the target task's state-space?
- How can good pairs of agents to function as source and target tasks be identified?

3.1. When to Transfer

In PTL deciding what to transfer to neighbours and when to do so is of utmost importance. There are many possible schemes to manage this. It could be driven by the target task requesting specific information, but any such scheme would be impractical because of the amount of computation and communication required for each transfer, so we will focus on source driven methods.

In RL the expected value of a state only become representative of the true value after this state has been experienced several times. From this, there are two approaches as to when the information about a state should be transferred. Information can be transferred every time it changes. The advantage of this is that there is little chance of the target task not having information when it has to make a decision about the state to which the information pertains. This method uses more bandwidth and there is a chance that what is being transferred may not be representative of the true value of a state as it has been only sparsely sampled. The alternate method is to only transfer information when a value appears to have converged (that being that the value changes more slowly than it previously did). This approach will use less bandwidth, has less risk of transferring inaccurate information but may not provide data in a timely manner.

No single method will be suitable for all states all of the time. The optimal frequency of transfer will depend on the frequency at which the target agent visits the state in question. It will also depend on how far into the learning process the agents are. At the beginning, information will be less reliable and as agents are encouraged to explore early on any information transferred will likely not be used to guide action selection, so, an initially low frequency of transfer would be best. As information becomes more reliable then the frequency can increase before tapering off when there is nothing left to learn.

3.2. What to Transfer

Related to determining the optimal frequency of transfer is selecting what data to transfer. Again there are several options; the preferable option in a particular system will depend on the nature and dynamics of that system. Transfer of the most recently visited states makes most sense as this information is freshest and is unlikely to have already been transferred. Sharing the most converged or most visited states are plausible schemes but they will tend to share the same information every time and in some systems agents will tend to have the same commonly visited states. This means that there will be little benefit in sharing that information. These schemes, however, will rarely pass on information received from other agents, so information will not propagate through the system particularly quickly. To ensure that transferred information gets shared quickly, an addition to the above schemes to promote propagation of transferred information can be used. Alternatively a scheme that transfers states that have experienced greatest change in values since last transfer to a particular target would accomplish this aim. Several of these methods for what to transfer and when will be analysed in the following section.

3.3. Receiving Information

Having identified suitable information to transfer and done so, the target agent needs to add this information to that which it already has. In this work, we will assume all agents can be trusted to only sharing properly formatted, correct information. The problem faced by the target agent is how best to determine if the received information is more accurate or better represents the environment than its current information. Maintaining statistics about how often a state has been visited and comparing this to how often the information received was sampled and accepting the most sampled value is a possible approach. It would, however, fail if the environment does not change the same way in response to each agent. This is particularly true if mapping from a distinctly different source and target, as there may not be a one to one correlation of convergence rates. This method will have little variation in the information it tends to transfer.

As only selecting either the transferred information or the local information will not permit agents to use multiple sources transferring information efficiently, local and received information will have to be merged. If merging the new information can move a particular value towards its true value, then it should be done otherwise the received information should be discarded. As the true value of a state cannot be known until the learning process is finished, estimates will have to suffice to determine if merging should occur. If there is no prior information for a given state then the received information should be accepted. The more difficult decision is merging if there is already information available. In this case the received information needs to be evaluated to determine if it is likely to be closer to the true value. Due to the way RL updates the value of a state-action pair, it will tend to move towards the true value over several visits to a state, this means that a direction of movement can be determined, this direction can then be used to see if the received information is in the right area for the true value. The steps towards the true value should get smaller the closer to convergence the value is, if the steps have already become small then it is probably not worth merging the received information. This merging scheme works reasonably well for the tested problems, however there is room for improvement. The information being transferred is not intended to provide the final exact value but to give an approximation so that fewer samples are needed to converge. This the significance attached to received data should drop as values approach convergence.

3.4. Source and Target Selection

With the mechanism to transfer information set, the agents must be able to determine which other agents may benefit from the information they hold. As the individual transfers are unidirectional, there are $N(N-1)$ possible transfers at any one time, where N is the number of agents. It is not feasible in terms of bandwidth or computation to have all of these transfers happen in addition to the communications the agents in the system are already executing nor is it scalable, so particularly useful transfers must be prioritised. Much like the other issues in PTL, knowing what is a good pair is not easily determined due to the fact that an agent's final, converged values are not known until learning has finished. This issue can be partially avoided by prioritising transfers that are directed to unvisited states. Transferring to unvisited states allows the agent to have prior knowledge about these states when first visiting them, so these transfers can be particularly beneficial. At a higher level, this means that an agent should preferentially share knowledge with agents exploring different parts of the environment than necessarily with its near neighbours. These pairs can be identified through a correlation function, where low correlation pairs will probably be good matches. This may not be feasible as it will require the whole state-space for all agents be available to each agent. A more practical approach is to look at the updates being shared in the multi-agent RL learning scheme, these will indicate what an agent is experiencing at a much lower cost.

As the converged source task is not available as is the case with sequential TL, PTL will see fewer benefits in terms of initial performance and learning time reduction. As parallelising TL can reduce the early effects of transfer, it needs to be determined which method is applicable for a particular system. This is a related problem to knowing if there is useful information to transfer. Obviously running both versions of the same system and comparing performance would solve this but in many large MASs this is infeasible.

4. Evaluation

This section will evaluate Parallel Transfer Learning. This will be done using a smart-grid scenario. The charging of electric vehicles (EV) will be coordinated by intelligent agents with one agent per vehicle. The experiments will be run using Distributed W-Learning (DWL) (Dusparic & Cahill, 2012) and GridLAB-D (Chassin et al., 2008). GridLAB-D is an electrical grid simulator developed by the US Department of Energy. DWL is a multi-agent RL method

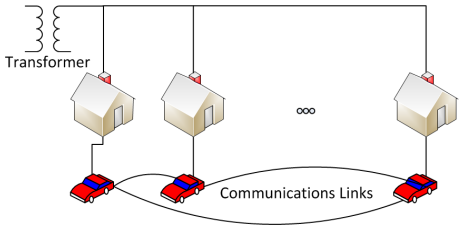

Figure 3. The smart-grid scenario used for these experiments.

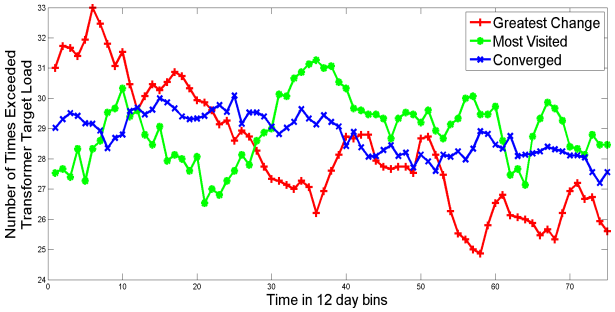

Figure 4. Moving average of number of times over desired transformer load using the Greatest Change, Most Visited and Converged methods.

that uses joint action learners. In joint action learner algorithms that use communication it is natural to apply Parallel TL as the agents are already sharing information about their state and reward scheme. The scenario covers one neighbourhood with 6 houses each with an EV (see Figure 3). The electric vehicles will need to coordinate their charging to prevent overloading the transformer while leaving sufficient capacity for the uncoordinated load from the houses (Taylor et al., 2012). They must also provide each EV with a sufficient battery charge to complete their daily journeys. The transformer is specified such that it can handle only two vehicles charging simultaneously. Each EV requires 5 hours charging to fully charge their battery. To charge sufficiently to meet their normal usage demand requires an average of 3.5 hours charging is needed. They are available for charging for 11.5 hours, this means it should be possible for each EV to have the required charge for all its journeys. As indicated before, there will be an agent representing each EV, they are capable of communicating with any other EV in the neighbourhood. At each time step they will receive a notification of the load on the transformer prior to their actions.

Performance will be measured based on two factors. The number of times the transformer is over its target load and number of times an EV returns home having run out of electricity (indicating it was not sufficiently charged prior to departing). The more rapidly good performance is achieved, the faster the agents are learning. Of particular interest will be early performance and improvement over the experiment. Two experiments will be run.

- Evaluation of differing data selection methods for Parallel Transfer Learning.
- Evaluation of Parallel Transfer Learning against Sequential Transfer Learning and no Transfer Learning.

The first experiment will compare different methods of selecting data to transfer. This will establish which of the discussed method of data selection is best in this scenario. The data selection methods used are (a) Greatest Change, (b) Most Visited and (c) Converged. The Greatest Change method will transfer which ever state has had the largest absolute change in value between two consecutive time steps. The Most Visited method will share the state that has been visited most often. The Converged method will select at random a state that has settled at a non-zero value, in effect it is transferring data that the agent is confident will not change dramatically as it has been well sampled. The experiments were run three times and for a period of 30 months.

Figure 4 shows the performance of three different methods of data selection. It compare three possible methods. The X-axis is the number of twelve day bins, the Y-axis is how many times in that particular bin did actual transformer load exceed target load. All methods show similar performance in terms of EV returning home completely discharging (i.e., not sufficiently charged prior to departure), with approximately 1 discharged EV every three days. As can be seen in Figure 4 the Greatest Change method shows poor initial performance. This is because in the early stages which ever state is visited is the state experiencing the greatest change. If a bad state-action with small positive reward is transferred to a state that has no other information, the action with information will tend to be selected at the neighbour. It takes time for this initial biasing to be overcome, once it is the performance steadily improves. The Converged method show a much slower improvement characterised by the gentler slope. As it waits for converged values to transfer, it has practically no effect until converged information is available. From this point some improvement is gained through transfer. Initially Most Visited performs best. This is because it is sharing information that has been sampled several time an therefore is more reliable. Its

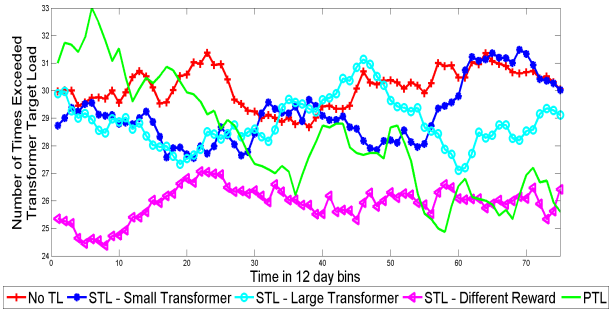

Figure 5. Moving average of number of times over desired transformer load using No Transfer Learning, Parallel Transfer Learning and Sequential Transfer Learning.

performance stabilities faster as the same pieces of information tend to be shared, this also inhibits its final performance as it does not find a final policy as good as other methods. This experiment shows that Greatest change is the best method evaluated here as it achieves superior final performance.

The second experiment will compare the performance of a system using no transfer learning, with that of both parallel and sequential TL. The sequential TL will be provided initial information from three different sources. All based on the same scenario but one with a smaller transformer, one with a larger transformer and one with a different reward structure but the same transformer.

The graph in Figure 5 shows a comparison of not using TL, using PTL and using STL with different sources tasks. The STL methods used are all given initial data from the end of a 30 month run. One uses a smaller transformer and the same reward structure, one uses a larger transformer and the same reward structure and the other one uses the same transformer and different reward structure. The final STL method will perform best as despite having different absolute values, their relative sizes will cause the same actions to be selected as were learned in the training run. The PTL system will be using the Greatest Change method to select data. All of the methods evaluated show similar EV charging performance when compared to the first experiment described before, so only transformer performance will be examined.

The early performance of PLT is, as expected, comparatively poor, it does significantly improve achieving final performance similar to that of the best STL method (the one trained on a the same scenario but with a different reward structure). It performs better than No TL after 7.74 months, this is when it overcomes the initial drop in performance due to using Greatest Change selection. It takes another 1.93 months to outperform STL using different sized transformers. This is the point information in the PTL system becomes more useful than the initial STL information. Catching up with the performance of these other methods demonstrates that the PLT scheme is learning faster than they are, this trend continues throughout its run.

The effectiveness of any TL system depends on how good the source information is. In the STL experiments with different sized transformers, the information was useful to a point but eventually the information available to the PTL system became more relevant. The STL information with a different reward structure had the best information to improve system performance and as a result it took significantly longer for PTL to match its performance.

These results show that PTL can be effective at improving the performance of a RL system but that it takes some time before it can outperform STL. As there is no requirement to have a source task prior to starting the system, there are benefits to offset the inferior initial performance.

5. Conclusions

To accelerate learning in a MAS this paper has proposed a new approach called Parallel Transfer Learning (PTL). Through allowing the source and target tasks to run simultaneously, agents in the system can learn from their peers' experiences. Previous work on TL had required that the source task be completed before the target task could begin. This necessitated transfer occurring only at the start of the target task's execution. By allowing transfer to occur throughout execution, information can be shared as soon as it becomes available. This means an agent can share information learned during execution not just what is available prior to commencing the learning process. It also takes advantage of the fact that, typically, agents in a MAS learn similar behaviours. Leveraging these experiences allows information to be utilised to its fullest extent, this being particularly beneficial in applications where individual experiences are expensive or rarely occur. The results show that PTL improves final performance and allows learning to occur more quickly once the initial cost of not having converged source information is overcome. PTL takes less time to reach a given performance level when compared to the time Sequential TL takes if the source task learning time is counted.

Acknowledgements

This research was supported by Science Foundation Ireland (SFI) under the Principal Investigator research program 10/IN.1/I2980 Self-organizing Architectures for Autonomic Management of Smart Cities and by SFI grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

References

Ammar, H. and Taylor, M. Reinforcement learning transfer via common subspaces. *Adaptive and Learning Agents*, pp. 21–36, 2012.

Ammar, H. B., Tuyls, K., Taylor, M. E., Driessens, K., and Weiss, G. Reinforcement Learning Transfer via Sparse Coding. In *AAMAS 2012: Proceedings of the eleventh international conference on autonomous agents and multiagent systems*, pp. 4–8, 2012.

Bianchi, R.A.C. and Bazzan, A.L.C. Combining independent and joint learning: a negotiation based approach. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1395–1396, 2012.

Boutsioukis, G., Partalas, I., and Vlahavas, I. Transfer learning in multi-agent reinforcement learning domains. *Recent Advances in Reinforcement Learning*, (Section 3):249–260, 2012.

Busoniu, L., Babuska, R., and De Schutter, B. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38 (2):156–172, 2008.

Chassin, D. P., Schneider, K., and Gerkensmeyer, C. Gridlab-d: An open-source power systems modeling and simulation environment. In *Transmission and Distribution Conference and Exposition, 2008. T&D. IEEE/PES*, pp. 1–5. IEEE, 2008.

Dimeas, A.L. and Hatziargyriou, N.D. Multi-agent reinforcement learning for microgrids. In *Power and Energy Society General Meeting, 2010 IEEE*, pp. 1–8. IEEE, 2010.

Drummond, C. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *arXiv preprint arXiv:1106.1796*, 2011.

Dusparic, I. and Cahill, V. Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):11, 2012.

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

Guestrin, C., Lagoudakis, M., and Parr, R. Coordinated reinforcement learning. In *Proceedings of IMCL 2002, The Nineteenth International Conference on Machine Learning*, pp. 227– 234, 2002.

Massoud Amin, S. and Wollenberg, B. F. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.

Nikos, V. *A Concise Introduction to Multiagent Systems and Distributed AI*. Morgan and Claypool Publishers, 2003.

Pan, S. J. and Yang, Q. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions onransactions on*, 22(10):1345–1359, 2010.

Powell, W. B. and Ma, J. A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications. *Journal of Control Theory and Applications*, 9(3):336–352, July 2011.

Stone, P. and Veloso, M. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

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

Sycara, K. P. Multiagent systems. *AI magazine*, 19 (2):79, 1998.

Taylor, A, Galván-López, E, Clarke, S, and Cahill, V. Management and control of energy usage and price using participatory sensing data. In *3rd International Workshop on Agent Technologies for Energy Systems (ATES), at AAMAS 2012*, pp. 111–119, 2012.

Taylor, M. E. and Stone, P. Transfer Learning for Reinforcement Learning Domains: A Survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

Taylor, M. E., Stone, P., and Liu, Y. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8: 2125–2167, 2007.