Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

OPTIMIZING EARTH MOVING OPERATIONS VIA REINFORCEMENT LEARNING

Vivswan Shitole
Joseph Louis
Prasad Tadepalli

Oregon State University
Corvallis, OR 97331-5502, USA

ABSTRACT

Earth moving operations are a critical component of construction and mining industries with a lot of potential for optimization and improved productivity. In this paper we combine discrete event simulation with reinforcement learning (RL) and neural networks to optimize these operations that tend to be cyclical and equipment-intensive. One advantage of RL is that it can learn near-optimal policies from the simulators with little human guidance. We compare three different RL methods including Q-learning, actor-critic, and trust region policy optimization and show that they all converge to significantly better policies than human-designed heuristics. We conclude that RL is a promising approach to automate and optimize earth moving and other similar expensive operations in construction, mining, and manufacturing industries.

1 INTRODUCTION

Earth moving operations that occur in the construction and mining industries require complex collaboration between multiple disparate and heterogeneous resources for their safe and efficient operation. In the recent past, productivity of both the above industries indicates a declining trend that threatens their well-being and necessitates novel means for optimizing operations. Construction, which is one of the oldest and largest engineering industries, has been noted to be particularly delayed in their adoption of novel automation and data analysis technologies, when compared to other similar industries like agriculture and manufacturing. Teicholz's study of construction productivities indicated a declining trend in its performance relative to non-farm industry sectors (Teicholz 2013) . Similarly, Durrant-Whyte et al. noted a global decline of 3.5 percent in productivity in the mining industry over the past decade (Durrant-Whyte et al. 2015).

While there are numerous reasons for the above performance declines in these industries, one of the most important reasons is the lack of efficient use of data in decision-making processes. It has been noted that the application of advanced computational techniques is critical to efficiently convert the data collected from the site into near-optimal decision making as it relates to field operations. The development of appropriate computational methods is especially important today given the growing ubiquity of sensors deployed on equipment that collect voluminous amounts of data, which in turn provides a tremendous opportunity for improving operational productivity when converted to actionable insight.

The current research is motivated by the need to develop advanced computational methods to utilize field-collected data to improve operations. Specifically, this paper focuses on combining discrete event simulations with model-free reinforcement learning (Sutton and Barto 2018) to identify optimal policies (Mnih et al. 2015) for earth moving operations. Reinforcement learning works by controlling the decisions of the simulated operation using a policy, which in our case is parameterized by the weights of a neural network. The weights of the neural network are updated online using different algorithms as the system controls the simulation and measures its own performance. In this paper we study three different algorithms including Q-learning (Sutton and Barto 2018), Advantage Actor-critic (A2C) (Sutton et al. 2000), and Trust Region Policy Optimization (TRPO) (Schulman et al. 2015). We show that all three algorithms

perform better compared to previously published heuristics. While the application focus of this paper is on earth moving operations in construction or mining, the developed framework can be readily applied to any operations characterized by multiple resources, high degree of uncertainty, and the need for long-range and strategic planning.

2 BACKGROUND AND RELATED WORK

This section introduces the reinforcement learning (RL) framework and summarizes its applications in other similar engineering domains. RL is a type of machine learning technique wherein an agent can learn a behavior policy to optimize an objective function (Sutton and Barto 2018). The objective function is expressed in terms of the "rewards" received by the agent, which expresses the direct benefits of an action. We adopt the commonly used objective function of expected discounted total reward, which geometrically discounts the reward received by the time it was received. The goal of RL is to identify an optimal policy that maximizes the objective function and thereby optimize the performance of system. Model-free Reinforcement Learning has the advantage that it does not presume an explicit probabilistic model of how the agent's actions effect the world, but instead learns good policies by interacting with a simulated environment.

One of the key elements of RL is a compact representation for describing the policy. Most RL approaches explicitly represent a so-called 'Q-function', which describes the desirability or value of an action in a given state or the 'value function', which is the desirability of a state. Since the number of possible states of the system is prohibitively large, these functions are represented compactly in a parametric form. While linear functions are often used for this purpose, recent work in Deep Reinforcement Learning (DRL) found that neural networks trained by gradient descent methods make very good function approximators with many successful applications including Atari games (Mnih et al. 2015) and the game of Go (Silver et al. 2016).

The capabilities of reinforcement learning have been noted to be particularly suitable in complex systems that involve uncertainty and for which a large number of state variables could affect performance. It has thus been successfully applied in numerous civil engineering contexts including transportation, water resource management, and dispatching of resources including trucks and elevator systems. In a survey of numerous applications of RL towards the optimization of traffic signals at intersections, it was found that RL can contribute significantly towards overcoming challenges in decision-making in the choice of traffic phase and durations for smooth traffic flow (Yau et al. 2017). One of these approaches implemented a multi-agent reinforcement learning framework to minimize queuing delays at intersections (Arel et al. 2010). RL has also been applied to compute optimal policies to regulate multi-reservoir systems by considering spatially and temporally correlated hydrological inflows (Lee and Labadie 2007). The typically high dimensional and stochastic multi-reservoir systems were provided with regulatory policies to optimize performance measures such as in-stream flow maintenance, and water supply for domestic, industrial, and agricultural uses.

The utility of RL has also been demonstrated in large-scale dynamic optimization problems such as elevator systems that operate in continuous space and time as discrete event systems (Crites and Barto 1996). A similar application for dispatching resources was implemented to sequence drainage trucks in marine container terminals to minimize their wait times (Fotuhi et al. 2013). An online preference learning system for dynamically adopting policies (Choe et al. 2016) was developed to dispatch automated ground vehicles in an automated container terminal, wherein their method was compared to RL. While the presented preference learning method was found to be a viable option for the problem domain considering the uncertainty involved, they did note that RL was very suitable option for real-time applicability.

To the best of our knowledge, this paper is the first to study the integration of RL with DES to provide a general framework for optimizing earth moving operations.

3 TECHNICAL APPROACH

The crux of our approach is to formulate a construction operation as a Markov Decision Process M and use Discrete Event Simulation E to simulate the MDP and obtain an optimal policy π^* which optimizes some objective defined for the operation. The policy π^* is theoretically guaranteed to be the optimal policy if we have enough samples from simulating the MDP and use a table-based representation of the Q-function and the value function.

3.1 Model-free Reinforcement Learning

A Markov Decision Process M is defined as the tuple (S, A, P, R, γ), where, S is a finite set of Markov states, A is a finite set of actions, $P : S \times A \times S \to \Re$ is the state transition probability distribution, $R : S \to \Re$ is the immediate reward function and $\gamma \in (0, 1)$ is the discount factor.

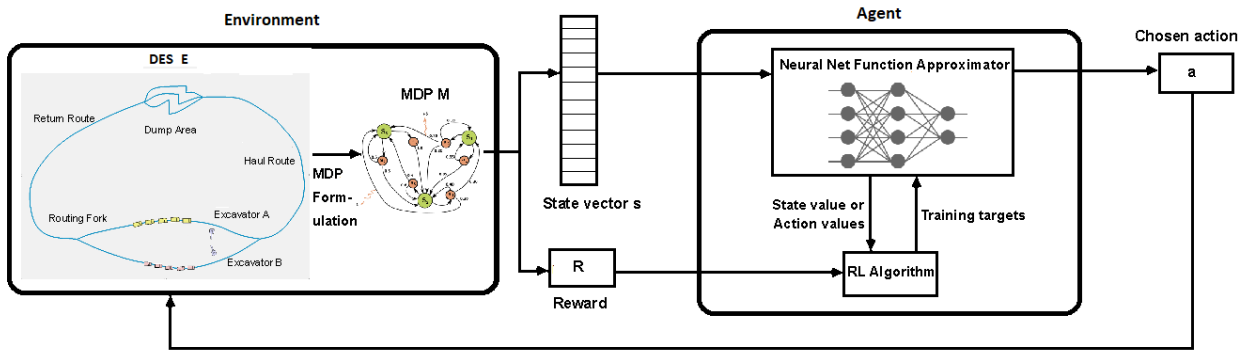


Figure 1: The deep reinforcement learning approach for solving Markov decision problems. On the left is a discrete event simulator which simulates the environment by taking actions on a state and producing the next state and the reward. The state vector is fed into the neural network function approximator, which proposes an action. The RL algorithms update the weights of the neural network based on the difference between the predicted long-term objectives in successive time-steps.

In a Markov Decision Process (MDP), all states follow the Markov property: $P(s_{n+1}|s_n, s_{n-1}, .., s_0) = P(s_{n+1}|s_n)$, i.e., the conditional probability of next state s_{n+1} depends only on the current state s_n and not on the sequence of states $\{s_{n-1}, .., s_0\}$ that preceded it. The solution to an MDP is a policy, which is a mapping from states to actions that optimizes a desired objective function. In general, policies are stochastic in that they map states and actions to the probability of taking that action in that state. The policies need to be stochastic to effectively balance the exploration-exploitation trade-off. Typically, the objective function is an expectation of the total reward received when following the policy. In problems with infinite horizon, the reward is geometrically discounted by the time at which it is received to encourage early accumulation of rewards and to keep the total finite.

In model-free reinforcement learning approach to solving an MDP, an agent (different from agent-based modeling) samples states from MDP M by interacting with environment E. The agent selects its action a from a stochastic policy $\pi(a|s, \theta) : S \times A \to [0, 1]$, parameterized by θ (as a linear function or a neural network). The environment E can be assumed as an oracle generating the next state s_{n+1} using an unknown (to the agent) transition probability matrix P, based on the agent's action a_n (obtained from agent's policy π) at the current state s_n. In addition to the next state s_{n+1}, the environment generates a reward $R(s_n)$ for the agent's action a_n at state s_n. Thus the agent's interaction with the environment generates a sequence of state-action-reward tuples, terminating at a terminal state s_T. This sequence $\{s_0, s_1, .., s_T\}$ constitutes an episode. The cumulative discounted sum of all rewards $\sum_{n=0}^{T} \gamma^n R(s_n) = G$ is the return from the episode.

Our goal is to find a stochastic policy π^* that maximizes the expected return G over all the episodes (samples) of the MDP experienced by the agent, i.e.,

$$\pi^* = \arg\max_{\pi} \quad \mathbb{E}_{s_0,a_0,\ldots,s_T,a_T}\left[\sum_{n=0}^{T} \gamma^n R(s_n)\right] \tag{1}$$

where, $s_n \sim P(s_n|s_{n-1},a_{n-1}), \quad a_n \sim \pi(a_n|s_n)$

3.2 Construction Operation as an RL Problem

A construction operation can be formulated as an MDP in the following manner: A construction operation can be viewed as a collection of recurring activities carried out together towards an end goal. A complete snapshot of the operation which captures entire information of the constituent activities at a given time would then define a markov state of the operation. The efficiency with which the activities progress together would form an objective (the return) to be optimized. The control variables used to control the activities would form the action variables. Different settings of these control variables would correspond to different policies that govern the efficiency of the operation. Simulating the operation on a Discrete Event System with different control parameter settings to see how the states and the objective unfolds is then akin to interacting with the environment using different policies to optimize the return. Hence our goal of learning the best settings of these control variables (best policies) to optimize the efficiency (objective) of the operation by simulating it on a DES (interacting with the environment) fits into the Reinforcement Learning framework over an MDP. Note that we favour simulation based RL approach to solving the MDP over non-simulation based methods such as mathematical programming or analytical fleet balancing for solving the MDP directly. The large number of possible states and the long horizon involved in the problem make it more difficult to solve the MDP using the direct approaches than using the simulation based RL approach.

3.3 Case Study: Earth Moving Operation

As an example to illustrate our framework, we consider a hypothetical earth moving operation as specified in (Louis and Dunston 2018). The operation requires the loading and transportation of material from a load area to a dump area by a fleet of trucks. Loading of the trucks is carried out by two excavators operating at two different load sites within the same general loading area. Loaded trucks then go to the dumping site via a common haul route. The dump site has an upper limit on the number of trucks that can unload simultaneously. The unloaded trucks then return to the loading area via a common return route. The DES in Figure.1 illustrates the operational model.

To formulate an MDP, the state of the system is represented by a (12×1) vector, with each element representing a different feature of the DES at a given time in the simulation. This state vector, as described in Figure.2, provides a complete snapshot of the DES at a given time, thus respecting the Markov property. The reward function is formulated as the negative of the time of cycle of a truck (the time taken by a truck to exit the routing fork, go through the excavator, the haul route, the dump area, the return route and arrive back at the routing fork). The presence of two excavators in the operation necessitates the routing fork as the decision making element which decides the excavator to which an empty truck should queue up for loading. Thus the routing fork constitutes the RL agent which has to select between the two actions in the action space (Figure.2) and learn an optimal routing strategy as its policy. The effect of different routing strategies on operational performance is measured quantitatively as the operation production rate, defined as the amount of material delivered to the dump site by the trucks over the time elapsed in the operation. The total time elapsed from the start of the operation (no material at dump site) to the end of the operation

(desired quantity of material at the dump site) constitutes the duration of a single episode. The production rate at the end of the operation constitutes the return G of the episode.

The outcome (final production rate) of this operation is non-obvious because of the involvement of many random variables which together introduce stochasticity in the operation. The two excavators used in the operation have different bucket capacities and cycle times. The fleet of trucks is comprised of two classes of trucks constituting different volumetric capacities and travelling speeds. The cycle times of the excavators (for the Load activity) and the trucks (for the Haul and Return activities) follow certain probabilistic distributions rather than being fixed numbers.

3.4 Reinforcement Learning Algorithms

We employed both value-function based RL algorithms (Q-learning) and policy based RL algorithms (Actor-Critic and Trust Region Policy Optimization) for learning the policies.

3.4.1 Q-learning:

The Q-learning algorithm (Sutton and Barto 2018) learns a target policy π indirectly by learning a Q-value function $Q(s,a,\mathbf{w})$ for the explored states as a function of the learnable parameters $\mathbf{w}$. The Q-value of a state-action pair (s,a) denotes the value of expected reward obtained by taking action a in state s and thereon following policy π for the rest of the episode. The update rules to learn the parameters of the Q-value for a state-action pair are given as:

$$\delta \leftarrow R + \gamma \max_{a'} Q(S', a', \mathbf{w}) - Q(s,a,\mathbf{w}) \tag{2}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \nabla Q(s,a,\mathbf{w}) \tag{3}$$

Thus, the Q-value updates for each explored state-action pair are calculated by first computing the temporal difference error (TD-error), i.e., the difference between the predicted Q-value from the previous state-action pair (s,a) and the value based on the immediate reward received and the current state's predicted best Q-value (Equation 2). The weights of the Q-function are then updated in proportion to the temporal difference error and the gradient of the Q-function with respect to the weights (Equation 3). α represents the learning rate.

With a converged Q function, the agent could always take the greedy action that maximizes the value of $Q(s,a)$ for every state s. However to make sure that the agent learns the correct Q values and does not prematurely converge to suboptimal values, it needs to take exploratory actions that ensure all state-action pairs are exercised during learning. Under conditions of sufficiently thorough exploration, the updates are theoretically guaranteed to converge to the optimal Q-value function Q^* obtained under the optimal policy π^*. This implies that we are guaranteed to find the optimal settings in DES that achieve the best efficiency for the construction operation, as long as all possible cases are encountered in the simulation and all possible actions are exercised.

3.4.2 Advantage Actor-Critic:

Policy based methods directly learn a parametrized function $\pi(a|s,\theta)$ for the policy. Actor-Critic methods, in addition to learning a parameterized policy function $\pi(a|s,\theta)$, learn a parameterized value function $v(s,\mathbf{w})$ simultaneously (Sutton and Barto 2018). The policy function $\pi(a|s,\theta)$ is used by the agent to take actions and explore the MDP. Hence it is termed as the 'actor.' The value function $v(s,\mathbf{w})$ serves dual roles: it is used to calculate a biased estimate of the return for the current state by bootstrapping from the values of next states, and it serves as a baseline for the estimated return. Hence the value function is termed as the critic. The update rules for the policy and value functions in the Advantage Actor-Critic algorithm are given as:

$$\delta \leftarrow R + \gamma v(s', \mathbf{w}) - v(s, \mathbf{w}) \tag{4}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_{\mathbf{w}}\, \delta\, \nabla v(s, \mathbf{w}) \tag{5}$$

$$\theta \leftarrow \theta + \alpha_{\theta}\, \delta\, \nabla ln\, \pi(a|s, \theta) \tag{6}$$

Equation 4 calculates the TD-error δ of the estimated value of the current state $v(s, \mathbf{w})$. Equation 5 represents a stochastic gradient update to the parameterized value function $v(s, \mathbf{w})$ (critic update) using the TD error δ and the critic learning rate α_w. Equation 6 represents a stochastic gradient update to the parameterized policy function $\pi(a|s, \theta)$ (actor update) using the TD error δ and the actor learning rate α_θ.

Most of the complex MDPs have very complex value functions, but may have simpler policy functions since they are a direct parameterization of actions to select for given states. Thus, policy-based methods such as Advantage Actor-Critic can easily converge to optimal policies for complex DES models, while value-based methods such as Q-learning will have a hard time exploring all states of the complex model in order to learn the complex value functions.

3.4.3 Trust Region Policy Optimization:

Trust Region Policy Optimization (Schulman et al. 2015) is an iterative method for policy optimization that ensures monotonic policy improvement by using a minorization-maximization (MM) algorithm. It optimizes a surrogate function representing a lower bound on policy improvement, subject to a trust region constraint between the new policy and the old policy:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right] \\ \text{subject to} \quad & \mathbb{E}_{s\sim\rho_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}}(\cdot|s)||\pi_\theta(\cdot|s))\right] \leq \delta \end{aligned} \tag{7}$$

Thus, the surrogate function to be maximized is an expectation over the parameterized Q-value function $Q_{\theta_{old}}(s,a)$ multiplied by an importance sampling weight of the current policy $\pi_\theta(a|s)$ over the action sampling distribution $q(a|s)$. The trust region constraint is that the KL divergence between the old policy $\pi_{\theta_{old}}$ and the current policy π_θ should be within a value δ (a hyperparameter). This ensures that the state distribution visited by the current policy is not too far from that of the old policy so that the value estimates can be trusted.

A major advantage of Trust Region Policy Optimization is that, unlike Q-learning and Advantage Actor-Critic, it guarantees a monotonic policy improvement on each update. Hence we are ensured to get a better policy for the construction operation on each iteration of the simulation.

3.5 Neural Network Function Approximator

We used a multi-layer fully-connected deep neural network as the function approximator to represent the parameterized value and policy functions. The network architecture is shown in Figure.2.

The first layer of weights $W1$ is shared between the value function and policy function networks. It is a (12×24) dimensional layer which takes the (1×12) dimensional state vector s^T as input and yields an intermediate (1×24) vector h as output. In the value function network, h is fed into a (24×1) dimensional layer $W2_v$, yielding the value $v(s)$ of the input state s. In the policy network, h is fed into a (24×12) dimensional layer $W2_p$, yielding a (1×12) dimensional intermediate output h_p. Next, h_p is fed into a (12×2) dimensional layer $W3_p$ yielding a (1×2) dimensional output o_p. Finally, o_p is passed through a softmax activation function to yield the action probabilities $\pi(a1|s)$ and $\pi(a2|s)$ for the two actions.

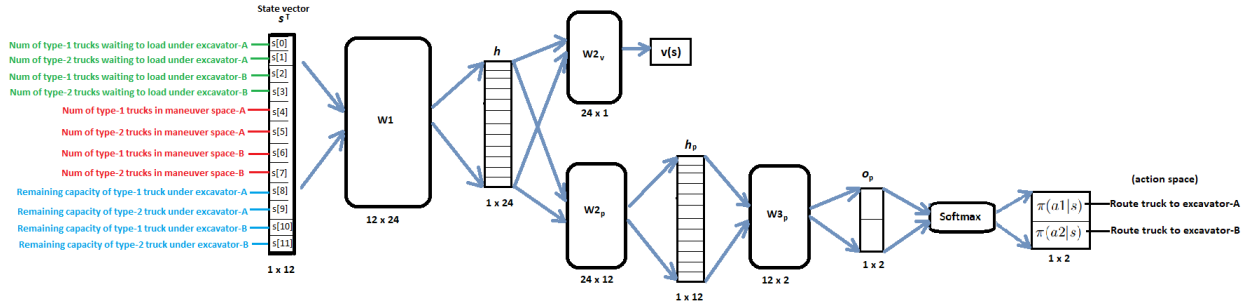


Figure 2: Neural Network architecture used to predict the value of the state and the action probabilities from state vector.

4 EXPERIMENTAL SETUP

Each experiment involved running the RL-integrated DES to learn a stationary policy for a fixed environment setting of the earthmoving operation. A fixed environment setting is characterized by a set of global parameters whose values were initialized as mentioned in (Louis and Dunston 2018) and kept constant throughout an experimental run. These parameters, along with their fixed values are given in Table.1. We used 200 episodes as the maximum number of iterations available to converge to a stationary policy. Each episode was started with the same initial state (equal number of trucks of each type under each excavator), the same initial weights of the neural network using Xavier initialization (Glorot and Bengio 2010) and zero amount of soil at the dumpsite. The episode is terminated when the desired amount of soil is dumped at the dumpsite. At the end of each episode, the episode length, the production rate and the unit cost of production are calculated as the performance metrics (Table.2). A policy is learned over the 200 episodes using the employed RL algorithm's updates, with Adam (Kingma and Ba 2014) as the optimization algorithm to update the weights of the neural network. The hyperparameters and exploration strategies used for each RL algorithm are specified in Table.3. In addition to learning and evaluating the performance of a stationary policy for a fixed environment setting, we ran experiments to compare the performance of policies learned over different environments. To achieve this, four different environments (scenarios) were created by using four different sets of values for the aforementioned global parameters that characterize a particular environment, followed by learning a policy for each environment. The global parameter values used in the four scenarios are listed in Table.4.

Table 1: Global parameters for default environment.

Global parameter	**Value**	**Global parameter**	**Value**	**Global parameter**	**Value**
num of type-1 trucks	5	type-2 truck capacity	3	desired soil amount	10000
num of type-2 trucks	5	excavator-A capacity	1.5	hourly cost for a truck	48
type-1 truck speed	15	excavator-A cycle time	Uniform[0.32, 0.34]	hourly cost for an excavator	65
type-1 truck capacity	6	excavator-B capacity	1.0	hourly overhead cost	75
type-2 truck speed	20	excavator-B cycle time	Uniform[0.32, 0.34]	simultaneous dump spots	3

Table 2: Performance metrics.

Metric description	Episode length	Production rate	Unit cost of production
Formula	$SimTime$	$DmpdSoil.level/SimTime$	$HourlyCost/ProductionRate$
Unit	hr	m^3/hr	$\$/m^3$

Table 3: Hyperparameters for RL algorithms.

Algorithm	**Hyperparameters**	**Exploration strategy**	**Discount factor**
Q-Learning	lr= 10^{-3}	ε-greedy with ε =0.1	0.99
A2C	actor-lr= 10^{-4}, critic-lr= 10^{-3}	added entropy of action probabilities to DNN training loss	0.99
TRPO	actor-lr= 10^{-4}, critic-lr= 10^{-3}, δ =0.003	added entropy of action probabilities to DNN training loss	0.99

Table 4: Global parameters of four different scenarios.

Scenario-1 (default)		**Scenario-2**		**Scenario-3**		**Scenario-4**	
Parameter	*Value*	*Parameter*	*Value*	*Parameter*	*Value*	*Parameter*	*Value*
Num of trucks	10	Num of trucks	10	Num of trucks	**20**	Num of trucks	10
Excavator-A capacity	1.5	Excavator-A capacity	1.5	Excavator-A capacity	1.5	Excavator-A capacity	1.5
Excavator-B capacity	1.0	Excavator-B capacity	1.0	Excavator-B capacity	1.0	Excavator-B capacity	1.0
Excavator-A cycle time	U[0.32, 0.34]	Excavator-A cycle time	**U[0.16, 0.17]**	Excavator-A cycle time	**U[0.16, 0.17]**	Excavator-A cycle time	U[0.32, 0.34]
Excavator-B cycle time	U[0.32, 0.34]	Excavator-B cycle time	U[0.32, 0.34]	Excavator-B cycle time	U[0.32, 0.34]	Excavator-B cycle time	U[0.32, 0.34]
Type-1 truck capacity	6	Type-1 truck capacity	6	Type-1 truck capacity	6	Type-1 truck capacity	6
Type-2 truck capacity	3	Type-2 truck capacity	3	Type-2 truck capacity	3	Type-2 truck capacity	3
Type-1 truck speed	15	Type-1 truck speed	15	Type-1 truck speed	15	Type-1 truck speed	**20**
Type-2 truck speed	20	Type-2 truck speed	20	Type-2 truck speed	20	Type-2 truck speed	**10**

5 EXPERIMENTAL RESULTS

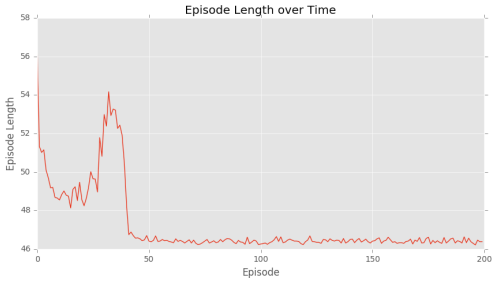
We evaluated the performance of Q-learning (QL), Advantage Actor-Critic (A2C) and Trust Region Policy Optimization (TRPO) for the given case study by generating two performance curves for each of the algorithms for 200 episodes based on episode length and episode reward. These results are shown in Figure.3.

As observed from the performance curves, all the three algorithms converge to approximately optimal policies. QL initially converges to a suboptimal policy and then diverges from it to converge to an approximately optimal policy (Figure.3(a) and (b)). This behavior can be attributed to insufficient exploration in the beginning and the local nature of gradient updates. A2C directly learns a policy function and hence is slightly more noisy in its learned parameters (Figure.3(c)). TRPO finds a better policy with every episode and never diverges due to its monotonic policy improvement guarantee (Figure.3(e)). It takes a slightly longer time to converge to an approximately optimal policy, as compared to QL and A2C due to the trust region constraint on its policy improvement step size. All three algorithms converge to similar optimal reward values.

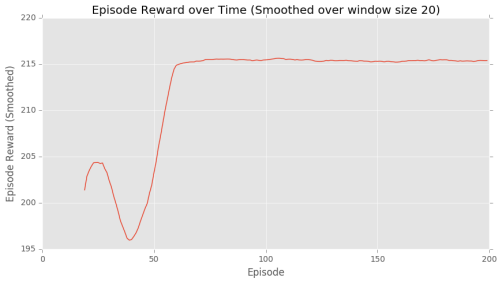
To check the robustness of the learned policies, we compared their performance with four heuristic policies (rule-based routing strategies) as described in (Louis and Dunston 2018). These heuristics prefer *Alternative Routing (H1)*, *Shorter Queue (H2)*, *Smaller Cumulative Capacity (H3)*, and *Earliest Load Time (H4)*. We carried out policy comparisons between the learned policies and the heuristic policies over four different scenarios generated by changing some of the environment parameters of the earth moving operation such as number of trucks, truck speeds, truck capacities, excavator bucket capacities and excavator cycle times. The obtained policy orderings are shown in Figure.4, where the first 4 correspond to H1 through H4, and the last 3 correspond to Q-learning, A2C and TRPO respectively. In all the scenarios, the learned policies converge to a higher reward (production rate) than the heuristic policies. This result implies that the learned policies are robust and perform better than the heuristic policies regardless of the environmental characteristics. On the other hand, there is no single heuristic that dominates the others in all scenarios.

We compared the performance of the neural network function approximator with a linear function approximator. Somewhat surprisingly, the linear function approximator did just as well as the neural network in most cases (not shown). To compare their efficacy in a more challenging domain, we repeated this experiment in an environment in which the excavators require periodic maintenance within every 30 minutes. This causes a maintenance downtime of 6 minutes for the excavator under maintenance. Forgoing any maintenance event within 30 minutes of the last maintenance event results in failure of the excavator.

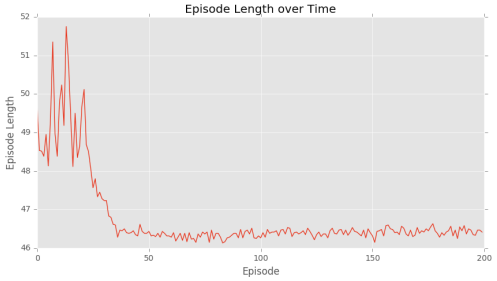
The failed excavator then needs to be repaired with a repair downtime of 2 hours. Hence the routing agent needs to come up with a policy that ensures maintenance of both the excavators, trading off the long repair downtime with short maintenance downtime. We employed the linear function and the neural network as the two cases of function approximator used by the routing agent to model the learnt policy in this setting. The curves depicting the production rates obtained for the two cases are shown in Figure 5. We see that the production rate on using neural network as the function approximator converges to a higher value than on using the linear function approximator. This implies that the neural network based function approximator was rich enough to learn a complex policy which trades off long repair downtime with short maintenance downtime. On the other hand, the linear function approximator based agent is not able to learn this complex policy and converges to a suboptimal policy that yields a lower production rate. Hence using the neural network as the function approximator over a linear function approximator is viable.



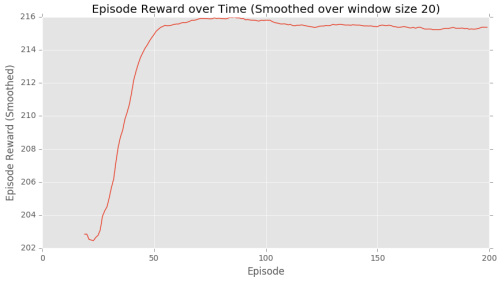
(a) Episode length: QL.



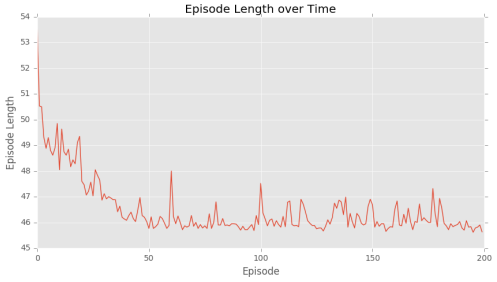
(b) Production rate: QL.



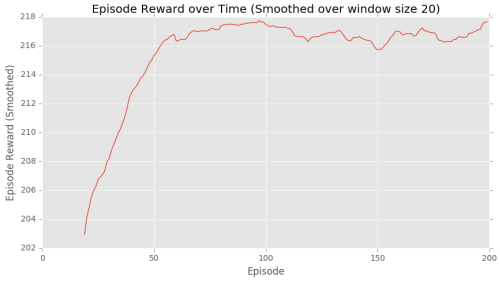
(c) Episode length: A2C.



(d) Production rate: A2C.

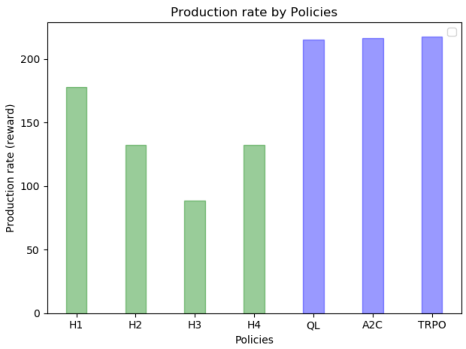


(e) Episode length: TRPO.

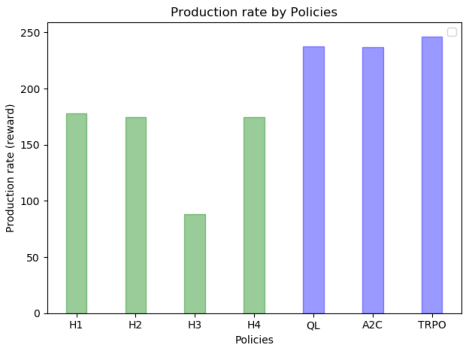


(f) Production rate: TRPO.

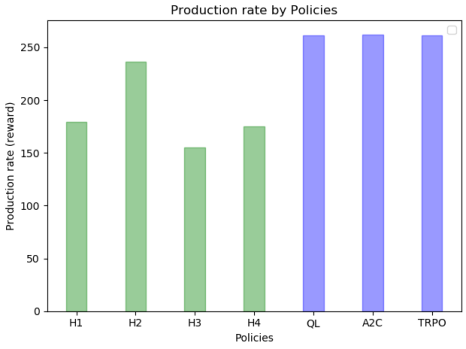
Figure 3: Performance of RL Algorithms employed: QL, A2C and TRPO for Scenario-1 (default). The X-axis shows the number of episodes. The Y-axis shows the average episode length on left (the smaller the better) and the average production rate on right (the higher the better).



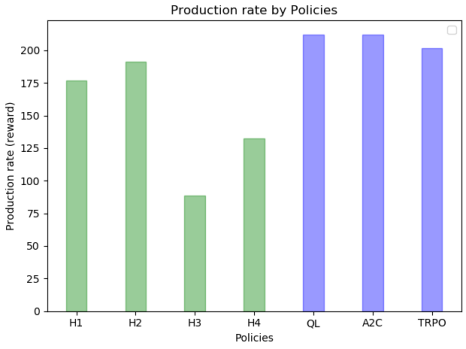
(a) Scenario 1 (default).



(b) Scenario 2.

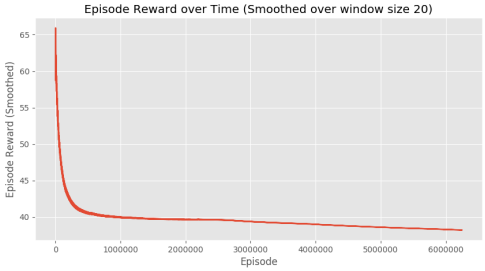


(c) Scenario 3.

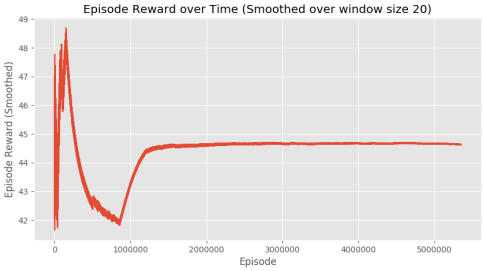


(d) Scenario 4.

Figure 4: Policy ordering for heuristic and learned policies. The first 4 (green) bars correspond to huamn-designed heuristics: Alternating, TrucksInQueue, CapacityInQueue, and AnticipatedLoadTime. The last 3 (blue) bars correspond to Q-learning, AC2, and TRPO respectively.



(a) Production rate: using linear function as function approximator.



(b) Production rate: using neural network as function approximator.

Figure 5: Curves depicting production rates on using the linear function and the neural network as the function approximator for modelling a complex maintenance policy. The production rate converges to a higher value on using neural network based function approximator implying a better (richer) learnt policy.

6 DISCUSSION AND CONCLUSIONS

We have shown that reinforcement learning in combination with discrete event simulation is an effective approach to optimizing earth moving operations, and performs better than previously published hand-designed heuristics. While the best performing heuristic changes from one scenario to another, the RL algorithms provide better performance in all scenarios analyzed. Deploying the RL algorithms in the real world necessitates speeding up the convergence of the algorithms, transferring the networks learned from simulations to the real world, and fine-tuning the networks using the real world experience. As a part of future work, we intend to move closer to these goals by extending our proposed method to a model-based RL approach from the model-free RL approach. Model-based RL approach involves exploiting the knowledge of the environment by building a model of the environment or using an existing model such as a simulator. This approach enables the RL agent to perform planning (learn from the model of the environment) before taking a real action and learning from interactions with the environment (as in the case of model-free RL). A limitation of our current model-free RL approach is that many samples of experience are required by the RL agent to learn an approximately optimal policy. Model-based RL approach is sample efficient as the RL agent can learn from planning, along with learning from real interactions with the environment. Another future extension of our current work is to include transfer learning, in which the RL agent leverages knowledge from previous environments and the policies learnt over those environments to learn an approximately optimal policy over the current environment in an efficient manner. We hypothesize that policies can be effectively transferred between environments that can be modeled by similar environment models. The policy over the current environment can bootstrap from the policy learnt over the previous (similar) environment and converge to an approximately optimal policy in much fewer experience samples than required in the case of learning the policy from scratch. Transfer learning is an essential research area for the deployment our proposed methodology to the real world since real world deployment involves transferring the policies learnt over the simulator to a different environment, i.e., the real world. Transfer learning can also be used to extend our current method to solve more expensive or complex operations in construction, mining and manufacturing.

REFERENCES

Arel, I., C. Liu, T. Urbanik, and A. G. Kohls. 2010. "Reinforcement Learning-based Multi-agent System for Network Traffic Signal Control". *IET Intelligent Transport Systems* 4(2):128–135.

Choe, R., J. Kim, and K. R. Ryu. 2016. "Online Preference Learning for Adaptive Dispatching of AGVs in an Automated Container Terminal". *Applied Soft Computing* 38:647–660.

Crites, R. H., and A. G. Barto. 1996. "Improving Elevator Performance using Reinforcement Learning". In *Advances In Neural Information Processing Systems*, 1017–1023.

Durrant-Whyte, H., R. Geraghty, F. Pujol, and R. Sellschop. 2015. "How Digital Innovation Can Improve Mining Productivity". *McKinsey & Company Insights*.

Fotuhi, F., N. Huynh, J. M. Vidal, and Y. Xie. 2013. "Modeling Yard Crane Operators as Reinforcement Learning Agents". *Research In Transportation Economics* 42(1):3–12.

Glorot, X., and Y. Bengio. 2010. "Understanding The Difficulty of Training Deep Feedforward Neural Networks". In *Proceedings Of The Thirteenth International Conference On Artificial Intelligence And Statistics*, 249–256.

Kingma, D., and J. Ba. 2014. "Adam: A Method for Stochastic Optimization". *ArXiv Preprint* 1412(6980).

Lee, J.-H., and J. W. Labadie. 2007. "Stochastic Optimization of Multireservoir Systems via Reinforcement Learning". *Water Resources Research* 43(11).

Louis, J., and P. S. Dunston. 2018. "Integrating IoT into Operational Workflows for Real-time and Automated Decision-making in Repetitive Construction Operations". *Automation In Construction* 94:317–327.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and S. Petersen. 2015. "Human-level Control through Deep Reinforcement Learning". *Nature* 518(7540):529.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz. 2015. "Trust Region Policy Optimization". In *International Conference On Machine Learning*, 1889–1897.

Silver, D., A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and S. Dieleman. 2016. "Mastering The Game of Go with Deep Neural Networks and Tree Search". *Nature* 529(7587):484.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Sutton, R. S., D. A. McAllester, S. P. Singh, and Y. Mansour. 2000. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In *Advances In Neural Information Processing Systems*, 1057–1063.

Teicholz, P. 2013. "Labor-productivity Declines in the Construction Industry: Causes and Remedies (A Second Look)". *AECbytes Viewpoint*.

Yau, K.-L. A., J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk. 2017. "A Survey on Reinforcement Learning Models and Algorithms for Traffic Signal Control". *ACM Computing Surveys (CSUR)* 50(3):34.

AUTHOR BIOGRAPHIES

VIVSWAN SHITOLE is a first year graduate student in the School of Electrical Engineering and Computer Science at Oregon State University. He is pursuing his Masters in Computer Science with specialization in Artificial Intelligence under the guidance of Dr. Prasad Tadepalli. His research interests include scalable reinforcement learning, multi-agent reinforcement learning and transfer learning. His email address is shitolev@oregonstate.edu

.

JOSEPH LOUIS is an Assistant Professor in the School of Civil and Construction Engineering at Oregon State University. His research interest lies at the intersection of simulation, visualization, and automation within the context of construction operations. He draws upon concepts in these areas to provide construction managers with better means of planning, monitoring, and controlling their operations to improve safety, maximize productivities, and minimize equipment idle times. His email address is joseph.louis@oregonstate.edu

.

PRASAD TADEPALLI is a Professor in the School of Electrical Engineering and Computer Science of Oregon State University. His main research interest is to understand learning and thinking by simulating them in computers. His work ranges from theoretical analyses of learning problems and algorithms to their implementation, evaluation, and application to real-world problems. He has co-authored over a hundred papers in artificial intelligence and machine learning in various journals, conferences, and workshops. He has organized many workshops and tutorials and co-chaired the international conference on inductive logic programming in 2007. He was a member of many conference program committees and is currently an action editor for the Journal of Artificial Intelligence Research, and the Machine Learning journal. His email address is tadepall@engr.orst.edu

.