Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture Complex Social Systems:
Modeling Agents, Learning, and Games

Project Report

Reinforcement Learning: Incentive Structure

Bryan Yu & Anya Heider & Anna Huwyler

Zurich
October 2020

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Name 1 Name 2

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1 Abstract

This report is about the project for the Complex Social Systems Lecture and it describes the problem we addressed, our approach and our results. In this project our team wanted to investigate how different reward schemes in reinforcement learning affects the agent behavior. We used the mini game "Mineral Collecting" from the game Starcraft II as simulation environment and implemented different reward schemes with the pysc2 library. The policy method algorithms we compared were A2C, DQN and PPO which we each combined with the different reward schemes. All results were compared to each other and to the baseline agent.

training period

The original idea was to additionally try a first attempt in multi-agent RL with our gathered results, therefore all decisions made in this project were made with that final goal in mind. Unfortunately, due to a changes in team composition, there was neither enough time nor manpower left in the end to implement the multi-agent RL as well.

2 Introduction

In society and other groups, common goals can interfere with personal goals of the individuals forming the group. In order to lead the group as a whole, politicians and other leaders therefore have to find ways to influence the people to act towards certain overarching group goals. Often, policy instruments are designed to give monetary incentives to influence the behaviour of single actors in a way that is favourable for the collective. A better understanding of decision making and the reaction towards different incentive structures could help to make more effective decision making. Computational social science and reinforcement learning as one method in this scientific field yield the potential to model learning and the adaption of agents' behaviour based on maximising the reward of single actors. As society as a whole is an overly complex system, we want to focus our research on a simpler example, where the effectiveness of incentives has proven useful.

Thus we begin with a motivating story from the annuals of business history of incentive driven behavior in a large American corporation[3]. In the late 20th century, Federal Express(FedEx), an U.S. business offered the general public overnight parcel delivery. As the story goes - every night FedEx would fly their planes from each major city into a centralized location (i.e. planes from New York, Los Angeles, Seattle, Miami, etc... would all meet in Chicago). Once all the planes congregated, employees would then re-organize packages from the source onto the appropriate planes according to the parcel's final destination. Thus, it is obvious the integrity of FedEx business depended on completing the delivery within the tight overnight window. However, as with most business stories, FedEx experienced difficulties meeting their goals as moving packages at the hub repeatedly took longer than expected. One day, someone in FedEx's management got the bright idea to pay the employees involved in the overnight delivery system not by the hour, but by that night's successful delivery. That is, once packages were completely loaded onto the destination planes, everyone can go home! The problem vanished - getting packages onto the destination plane was no longer a bottleneck.

As a group, we hypothesized that if the story's outcome resulted from incentive driven decision making, then similar behavior could be simulated using computational methods – specifically reinforcement learning and the reward optimizing agents within. For us humans these optimisations are quite easy to do, but how would a computer try to optimize this problem. Furthermore, if the results obtained were driven from changes in incentive, then would it be reasonable to expect to be able to observe similar behavior in reward optimizing agents? The following chapters will show our attempt to replicate this dynamics with Reinforcement Learning.

3 Hypothesis

We hypothesize that incentive caused behaviors can be modeled by computational methods.

In particular, we believe reinforcement learning, with it's design of using feedback signals to teach an agent about an unknown environment, serves as an interesting framework with minimal assumptions in which to explore our hypothesis. That is, we do not explicitly program the agent to take a course of actions. Instead, by defining a set of actions (up,down,left,right) and providing a reward signal, we expect to see the agents learn to make decisions which exhibit behavior aligned with the incentive structure.

In the context of the case study, the modified reward structure which allows employees to go home upon completion of the night tasks incentivizes the employees to finish sooner. Similarly, within the context of our mineral collecting agents, we believe the time incentive and the mineral thresholding will incentivize the agent to collect the minerals sooner.

A a baseline, we introduce a ”random agent” as an automata programmed to take one of the four actions (up, down, left, right) uniformly at random at every step. Furthermore, we consider this to be a model of the least incentivized agent in which to compare our results.

If our hypothesis is correct, we expect at the completion of the experiment to have trained agents who are self-incentivized to maximize the rewards our environment gives them. Quantitatively, we expect to see a increase in mineral collection of our agents as compared to the random agent beyond statistical chance. Furthermore we hypothesize that it might be possible to identify an increase in mineral collection as caused by changes in incentive structures. In other words, we hypothesize that introducing a time incentive will lead to larger mineral collected earlier in a episode.

However, as identified by the course teaching team, we believe this experiment may fail due to the computational requirements required to obtain a well trained agent. This computational requirement may detract from focusing on the issue of complex social systems. As a team, we respond to this concern with the simple note - ”Just because it is hard doesn't mean it is not worth trying.” We believe our experimental design is grounded in logic, reasonably designed, and worth attempting.

Furthermore, our initial interest in the multi-agent regime arises from the hypothesis that it should be. possible to capture emergent co-operative behavior driven solely by incentive structures. That is, in the single-controller reinforcement learning regime, one controller manipulates the two agents. As such, all rewards are attributed to the single controller. We hypothesize that in the multi-agent regime, allocating one self-serving controller per marine would lead to different behaviors as each agent will be attributed a portion of the entire environmental rewards and as such, their decentralized decision making will be driven by their immediate, personal, rewards.

Furthermore, we hypothesize that changes in reward structure would have more prominent effects. For example, if rewards were not shared between the agents(i.e. winner-takes-all), then it the agents were spawned near each other, we might fairly expect adversarial behavior such as where the agent learns behavior which actively pre-empt the other agent from collecting minerals – by taking it first. As such, the pre-empting agent will capture all the rewards of the episode.

Unfortunately, due to changes in team composition, we faced neither neither time nor enough manpower to investigate the multi-agent RL regime. We leave this hypothesis for future studies.

4 Theory

In the following chapter the theories and the calculations behind our simulations can be found. The first section describes the basics of reinforcement learning and the methods used in the simulations. The last sections should give a overview of the implementation, visualisation and measurement of the simulations.

4.1 Reinforcement Learning (RL)

Reinforcement Learning is commonly used in the context of decision making and learning an optimal behavior. Within an environment, an agent makes a series of sequential decisions and from the interaction with the environment, receives the a series of rewards. With help of a suitable reward systems the agents learns and tries to maximise the total reward based on the different actions taken. (see Figure 1)

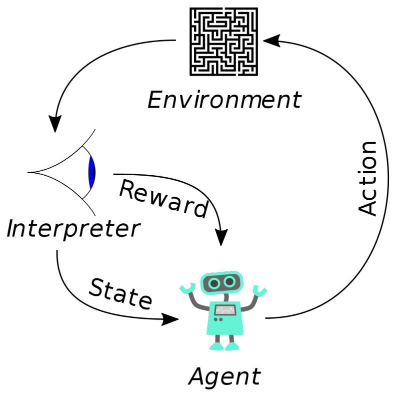


Figure 1: Reinforcement Learning [7]

The RL agent tries to maximize the reward system usually with help of the Markov Decision Process: $R_t = \sum_{k=0}^{\infty} \gamma_k r_{t+k}$, where r_t is the reward at time instance t and $\gamma_k \leq 1$ is the discount factor (which weights the rewards accordingly). The value of a state is the expected/mean value of the return R_t in a given state, so $V(s_t) = E[R_t|s_t]$. The state value is sometimes called the baseline. The Q-value, $Q(s_t, a_t) = E[R_t|s_t; a_t]$ is similar but considers the state-action pair. Subtracting the state value from the Q-value results in the advantage value $A(s_t; a_t) = Q(s_t|a_t) - V(s_t)$ and it serves a metric of how much better it is to take a specific action a_t compared to the average/general action at a given state s_t. The probability distribution π of selecting an actions a_t given a state s_t is called the policy: $\pi(a_t|s_t; \theta)$. Policies are parameterized by a set of variables θ, which are used to optimize the overall reward $J(\theta)$ over the trajectory τ: $J(\theta) = E[\sum_{t=0}^{\tau} R_t; \pi_\theta]$ An optimal reward

$J(\theta)$ can be located when the derivative is equal to zero: $\frac{d}{d\theta}J(\theta) = \nabla_\theta J(\theta) = 0$. This term calculated for the $\nabla_\theta J(\theta)$ is called the gradient step and there are different ways of doing so.
In the following paragraph the focus is laid on three specific policy gradient methods: A2C, PPO and DQN.

Deep Q-Networks (DQN)[2] is one reinforcement learning algorithm popularized through its success in the Atari environment. DQN introduces two major changes to the traditional reinforcement learning architecture. Firstly, DQN applies a convolutional neural network to teach the agents to learn the control policies (as compared to tabular methods). Secondly, DQN introduces experience replay, a method which alleviates the problem of correlated data and non-stationary distributions by randomly sampling from a pool of previous transitions.[2] An epsilon greedy strategy is applied which results in the agent not always taking the most probable actions but also occasionally some random actions. 2). [6]

As DQN is a milestones of modern reinforcement learning, our group decided to include it in the repertoire of techniques in our experiments.

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \bigg)}^{\text{temporal difference}}$$

Figure 2: Q-Learning update rule

Advantage Actor Critic (A2C) is a synchronous implementation of A3C[1]. A3C's initial contribution was the idea of multiple workers in individual environments running in parallel but the updates would affect a shared model. [1]. As a result, the risks of correlated data and non-stationary distributions could be alleviated without incurring the computational expenses of experience replay. Furthermore, A3C provided an additional contribution by demonstrating that similar levels of learning could be achieved without specialized hardware and in less time. Further research showed that the synchronous version achieved similar performance and as such, A2C is thus the synchronous version of A3C. That is, the update waits until every actor has completed its segment of experience before conducting the update step.

A2C aims to estimate the state value $V(s_t|\theta_v)$ with help of the parameter θ_v (Critic) and uses the gradient step as $E[\nabla_\theta \log(\pi(a_t|s_t;\theta_v))A(s_t,a_t;\theta,\theta_v)]$ with $A(s_t,a_t;\theta,\theta_v)$ being the advantage function. The gradient update is is applied after a fixed length duration has passed or if a terminal state is reached. A2C also parameterizes the policy $\pi(\alpha_t|s_t;\theta)$(Actor) with the parameters θ. While θ and θ_v are displayed as separate parameters, they can be shared for improved efficiency.[1].

Our group selected A2C as part of our experiment to take advantage of the supposed learning efficiency. We hypothesize that including this approach might give us insights to whether insufficient learning duration may be hindering the agent's learning meaningful policies and will discuss the social implications below.

Proximal Policy Optimization (PPO) [4], a policy optimization method, takes advantage of the multiple workers in A2C and also introduces a trust region used to improve the actor. The main idea in PPO is that modifications should not cause too much differentiation from the existing policy. As such, a trust region is applied which constrains on the size of the updates.[4]. Furthermore, Schulman et al[4] proposed the L^{CLIP}(equation 1 objective function where ϵ is a hyperparameter, θ are the policy parameters, and $\hat{A}$ is the advantage function. Equation 1 results in a pessimistic bound on valuations.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{1}$$

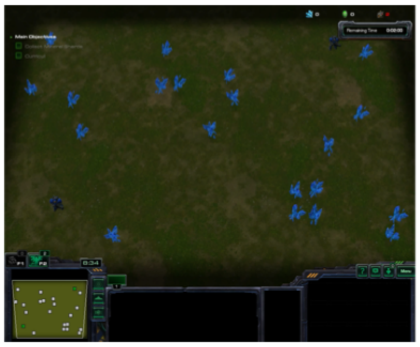
While DQN has shown success in the discrete action space, it had challenges when faced with continuous control. PPO is expected to improve the performance in the continuous control space. Thus, we include PPO into our repertoire of algorithms to (1) see how the triple advantages conferred from trust regions, multiple workers, and continuous control benefits may affect the agents' learned behaviors (2) investigate the implications of restricting changes when attempting to optimize a process (i.e. fedex system).

4.2 Choosing A Task and Environment

It is beneficial to find an already existing environment with an easy access to the source code and possible features for the reinforcement learning agents. After some research on different environments and minigames, the team decided to implement the simulations in a already existing mini game called "Collect Mineral Shards" inside the "StarCraft II Learning Environment"[5]. In this Mini Game there are two agents in the environment and their goal is to navigate the map and collect the minerals.

This game has the advantages of an already existing library called pysc2 which allowed the experiment to use a standardized environment, allow for ready experiment replication, and allows the team to focus on the course concepts and not on programming and software technicalities.

Figure 3 displays the mineral collection environment. The environment by default spawns two marines and twenty mineral crystals. Each of the marines can be controlled by the agent. When the marines walk on top of a mineral crystal, as per figure 3 (b), the mineral will disappear from the map and the mineral collection score will be incremented by 100. Each simulation is allowed 2 minutes which translates into approximately 240 actions to be taken. On every step of the single-agent reinforcement learning controller, the agent can select one of two marines and one of four directions (up,down,left,right) to move. Upon termination of an episode determined by the 2 minute timer expiring, the environment is reset, the mineral collection counter set back to 0, and the marines and minerals are re-spawned on the map randomly.


(a) State of the environment on the initialization of a new simulation

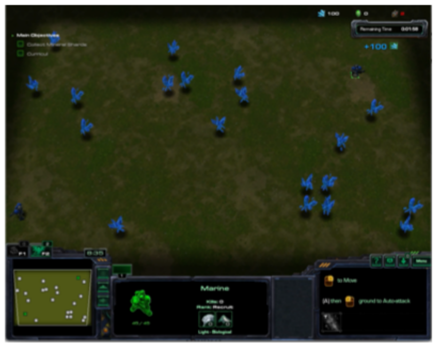

(b) The agent retrieves one mineral resulting in a +100 mineral collected score

Figure 3: Visualisation

4.3 Measurement and Comparisons

We'll be testing under the incentive and non-incentive scheme. we'll compare to the standard "Random agent" which picks a random action every step.

The blue nodes in Figure 4 are symbolized minerals and their placement on the game map. The shorter the path is that the agents has to take in order to collect the mineral, the faster he gets finished with the task and therefore the problem can be represented as the "Find the shortest path problem". Usually to find the optimal solution this task would take years for a computer to calculate. But with help of the reinforcement learning we can speed up this process a bit and at the same time be open for new maps. shorter path is equivalent with more mineral in less time. So if the result show that the agents collect more minerals in same time. red)agent chard

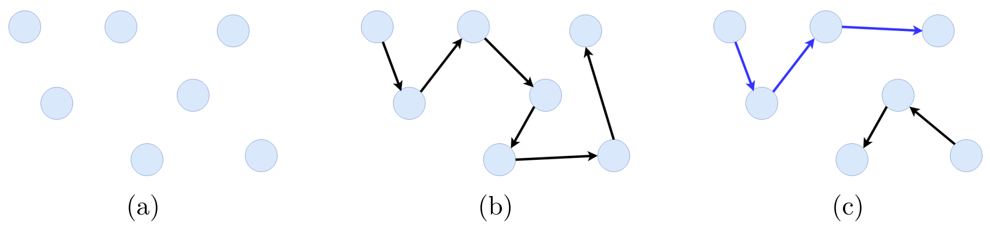


Figure 4: Find the shortest path

spacebetter and found a better solution to the shortest path problem and divided the maps between the two agents in a productive way.(In the Figure 4 (c) a possible outcome of the two agents dividing the maps[the path of one agent is blue the other is black]) and therefore the agents a re working together to achieve the goals of collecting as many minerals in time as possible.

Therefore, we measure the amount of minerals the agents pick up after ... (time) The asymptotic function for a two-dimensional random walk as the number of steps increases is given by a Rayleigh distribution. The probability distribution is a function of the radius from the origin and the step length is constant for each step. $P(r) = \frac{2r}{N}e^{-r^2/N}$
start anywhere in the grid, then move for a number of steps V log V, then your close to the uniform distribution i.e. a 1/V chance of being at any of the vertices in the disk of area R^2

5 Experiment

We describe our experimental design below.

The purpose of our investigation is to see whether, simply by changing the reward structure – agents who focus on maximizing their personal interests(rewards) will learn and demonstrate new emergent behavior.

We will thus explore three algorithms (A2C, PPO, DQN) as described above. Furthermore, we will introduce three reward configurations.

Baseline Agent: For our baseline agent, we introduce a random action agent. That is, this agent will take a random action in every timestep.

Time Incentive introduces a penalty(negative reward) to agent for every action it takes. In theory, this reward design would incentivize the agents to gather as many minerals as possible with as few steps. However, we note that it may be possible to collect all the minerals on the screen without running out of time and as such, the agents can incur additional penalties from having nothing to do.

Mineral Thresholding is thus introduced such that the time incentive rewards are terminated after a thresholding mineral has been collected. Afterwards, all minerals collected contribute the full 100 value to the agent and additional steps incurs no additional costs.
The group relates this design to the case study in that once the task at hand has been completed, the workers can go home. We note that we can improve on this design by terminating the episode and resetting the environment upon the collection of a threshold amount of mineral. We leave this to future experiment design.

Episodic Reward makes it such that the agent only receives reward indicators at the end of an simulation. That is, throughout the simulation the agent will get zero feedback. Upon completion of the simulation, the agent will receive one feedback signal.
Episodic rewards simulate the difficulties of communicating intention with workers and serve as a baseline in which incremental rewards can be compared. When the episodic reward configuration is not applied, the environment will give the agent continuous feedback after every action taken.

Experiments: We conduct 24 experiments by applying a combination of the three configurations above to each of the three algorithms. Every experiment trains the algorithm for 10^6 time steps. Subsequently the trained agent is ran for 10 episodes and the amount of mineral collected is tabulated. Furthermore, the baseline random agent will be ran for 10 simulations to present a baseline in which to evaluate the performance of the learned agents.

Expectations We expect that A2C will perform the best of the three algorithms due to it effectively distributing learning across multiple simulations without restricting learning updates. That is, we hypothesize this would be similar to having multiple hubs sharing knowledge and that the best ideas can be implemented effective immediately – not amortized in over time.

We hypothesize that the incentive structures will lead to the agents learning more optimal behavior (i.e. less random wandering, and more direct movement towards minerals). However, we cannot determine whether the training time we allocate will be sufficient for this behavior to be learned as previous literature report extended training periods on specialized hardware.

With regards to episodic rewards as opposed to continuous rewards, we believe that providing continuous feedback(rewards signals) will result in an improved agent. This expectation arises from the real world analogy of a teacher who gives continuous feedback to the student and that of a teacher whose only response comes at the end of a period. (i.e. multiple graded assignments versus one final). Though it is not obvious which design will work better.

The simulations were ran on a private server on Google Cloud Compute as well as the Euler Cluster(though configuring the bash script to run on a compute node was not finished in time).

6 Results

This chapter presents the results obtained from the experiments. It is further split into results from the training period presented in chapter 6.1 and final results comparing the performance of different model and incentive combinations in chapter 6.2.

6.1 Training Period

In order to evaluate the performance of the different algorithms, the development of the overall reward is extracted over the interval of training. Figure 5 shows exemplary developments of the reward over the training episodes. As the outcome of different runs are highly influenced by stochastic behaviour, respectively 10000 decision processes and the rewards at the respective points in time are summarized to one episode. This way, we hope to obtain a clearer visible trend within the measurements.

The left plot shows the development of rewards for PPO, A2C and DQN for the training with mineral thresholding incentive. The course of reward development is visualised in a line plot with the confidence interval of 95%. This example shows some overall trends that could be observed throughout the different training sets. A first observation is the difference in conducted steps or decisions that the different algorithms undertake during training. The time of training was restricted, so the vaying number of episodes has to stem from different time expenditures for the algorithms to take single actions. As seen in the figure, DQN depicts the lowest number of episodes, followed by PPO while A2C seems to tend to the fastest decisions. However, it has to be mentioned that in single other experiments, the order is found to be reverse. It is therefore only an overall trend that does not hold for the entirety of runs. Another observation is that the training period is probably not sufficiently high. For both DQN and A2C no clear trend of development is visible. PPO show an upward trend, though it is not clear if with further training this trend would continue.

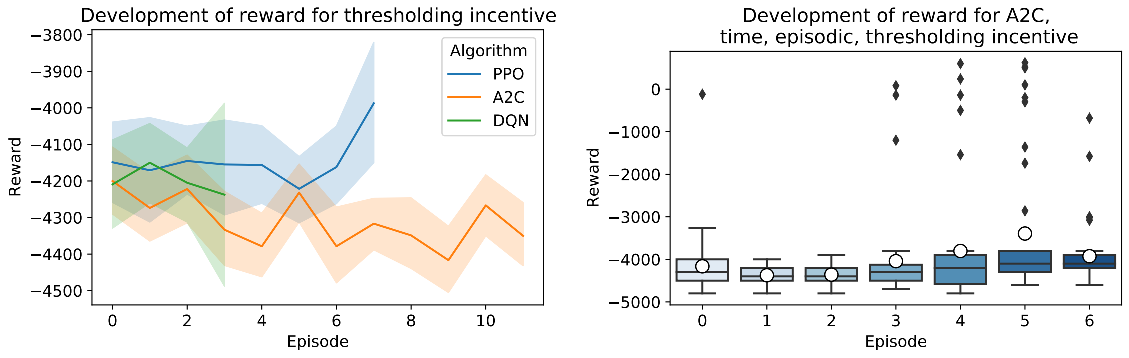


Figure 5: Exemplary development of the rewards during training.

The right side of the figure shows the development of incentives gained for the training of A2C with time, episodic incentives and mineral thresholding. In this case, another visualisation option is chosen in order to depict an interesting effect. The box plots allow an statistical interpretation of the data, also depicting outliers which are discarded in the representation of the line plots. The white circles in the figure display the mean. From the values it can be seen that the negative reward resulting from the time incentive lies at around -4900, higher values result from collected minerals. However, from the final results, it is clear that the maximum amount of collected minerals lies at 1800. Therefore the reward being close to 0 or even reaching positive values towards the end of the training interval is likely to result from the agent becoming faster in obtaining the threshold amount of minerals and stopping the negative time incentive at an early stage.

6.2 Model and Incentive Comparison

After the training period, each of the models is evaluated by running the experiment ten times and comparing the results. Figure 6 shows the development of accumulated reward due to collected minerals over time. For each configuration, the trained models are ran ten times to account for the probabilistic nature of the setup. The subplots of the figure show the outcomes for the three different algorithms for one specific incentive scheme, respectively. The configuration of the different experiments and their results can also be obtained from table 1 in the appendix.
The results show that DQN overall shows the worst results. The reason for this low performance could be the slower decision making observed during the training periods. With less episodes of training the algorithm has less opportunity to actually learn and therefore the results remain comparably low. For the default setting without further incentives, PPO seems to perform best. For all configurations including

episodic reward, PPO and A2C seem to perform similarly well. For time incentive, mineral thresholding and the combination of the two, A2C seems to outperform the other two algorithms.

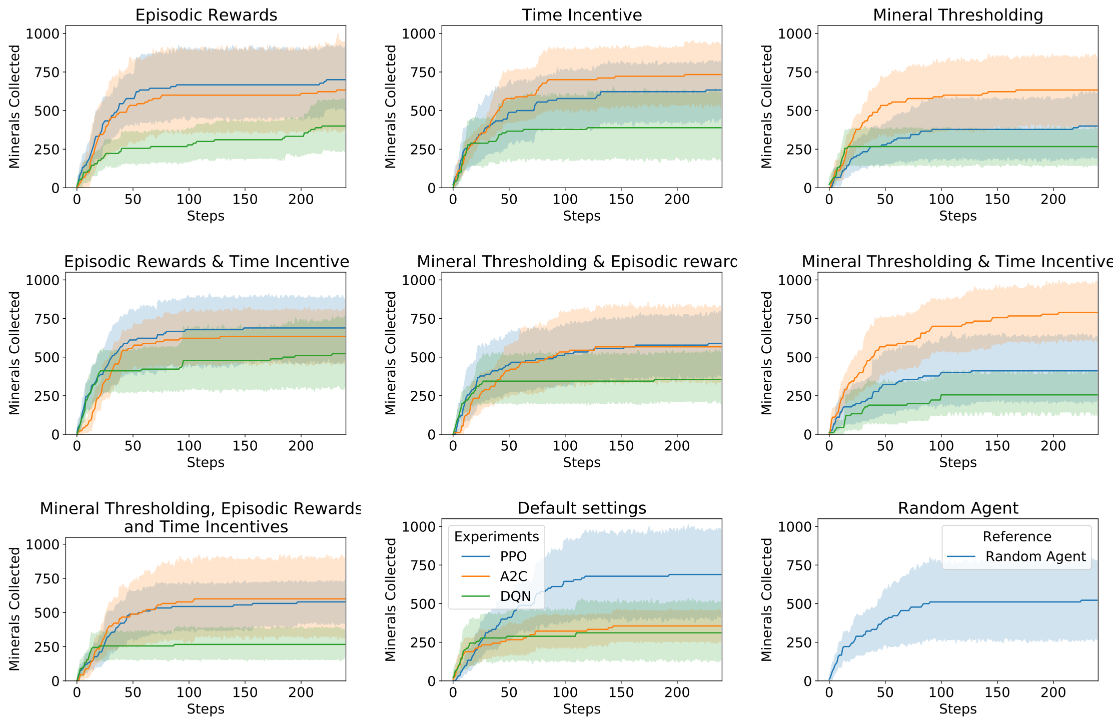


Figure 6: Comparison of accumulated rewards gained by collected minerals with different experiment settings.

6.3 Discussion of the effects of algorithm learning choices

We consider the implications of design of the algorithms chosen. As the random agent receives zero training, we consider DQN as the baseline for which to interpret the effects of A2C and PPO.

As per fig. 6, we see that DQN consistently under performs. Considered in relation to A2C and PPO – which uses simultaneous workers – we reason that the result we see may be driven from under-training the agent. DQN only has one agent operating at any time, and it may be that 1e6 training steps is insufficient for the agent to learn any meaningful policy. That is, we connect this observation to our case study by considering the situation in which only one worker without any prior knowledge, is tasked to improve the system. As such, we believe it is reasonable to see the worker perform poorly. Furthermore, we also note that DQN cumulative rewards appears to flatten out sooner. This behavior suggests that the agent does not collect minerals beyond its immediate vicinity. We reason it demonstrates that the agent is confused. Nonetheless, this demonstrates that insufficient or bad learning can hinder the development of a social system and the ability to accomplish tasks.

We note that PPO and A2C demonstrated improved performance over the random agent. We reason this performance improvement comes from the additional training resulting from instantiating multiple workers who simultaneously update a shared model. In the context of [3], we analogize this to having multiple hubs (i.e. USA, Europe, Asia, etc) who share best-practices with Headquarters with which the other hubs has ready access to. As a result, the system as a whole gets to benefit from the experiences of all its branches, and not just one site. However, we also note that A2C and PPO show tradeoffs in top performance across the incentive schemes. We reason one of the driver of this result is the clipping mechanism inherent in PPO, which restrict updates to the local vicinity. In the context of [3] or improving a business process, this would be analagous to a manager amortizing in a new best-practice – slowly and overtime. In contrast, A2C allows for any gradient update depending on the experience of its agents. That is, if the going gets good, then things can really take off! We believe this is what drives the large gap we see from A2C and PPO in "Mineral Thresholding" and "Mineral Thresholding Time Incentive."

We find "Mineral Thresholding Time Incentive", "Time Incentive", and "Mineral Thresholding" particularly interesting.

6.4 Mineral Thresholding

To evaluate the influence of the mineral threshold, we chose the two values of 600 and 1000 for experiment numbers 16 and 23, representing PPO with time incentive and mineral thresholding and A2C with episodic, time incentives and mineral thresholding. The decision for 600 was determined from the random agent's average performance. Figure 7 shows the results of this variation. For A2C, there is no visible effect observable. For PPO on the other hand, a performance increase can be observed for the lower threshold value.

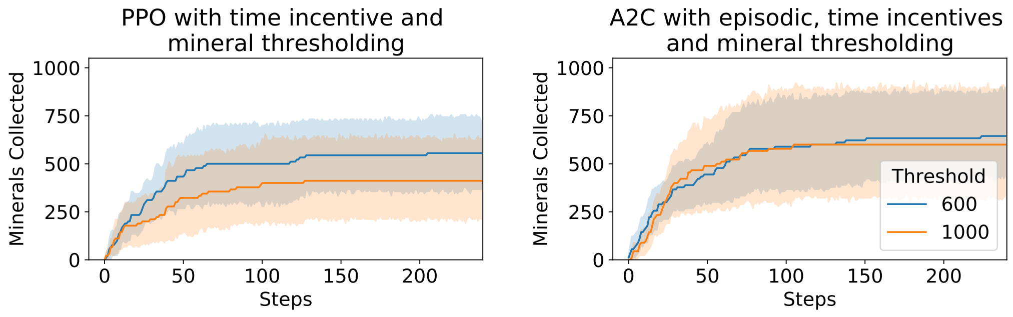


Figure 7: For the configuration of PPO, we see an improvement of performance with a lower threshold while for A2C there is no visible difference.

6.5 Time Incentive

From our experiments (fig 6), we noted that time incentives generated the most promising results of the batch. That is when ran as the only incentive modification, A2C outperforms the random agent. We note this result is in line with our hypothesis, that faced with a negative incentive, the agent can only maintain a previous reward level by learning to collect minerals. However, it is interesting to note that the time incentive did not seem to lead to the cumulative minerals collected curves from shifting leftwards – signifying that the agents learned to do the same task faster. For example, compared to ”Episodic Rewards”, ”Time Incentive” also appears to reach 500 minerals collected at around 50 steps under A2C. Nonetheless, the incentive is a step up from the random agent, who reaches 500 minerals at approximately 100 steps.

6.6 Mineral Thresholding and Time Incentive

Our most promising result arose from the ”Mineral Thesholding and Time Incentive” experiment in which A2C achieves a cumulative episode reward of over 750 minerals. In this scenario, agents are incentivized to collect minerals sooner, and after passing a threshold, are no longer penalized for further collections. Within the experiments here, the agents were faced with a threshold at 1000 minerals, and as such would have faced the same negative incentive in both ”Time Incentive” and ”Mineral Thresholding Time Incentive” as the 1,000 mark was not passed on average. Thus, we reason that in the training, it may have been possible the agents broke the 1,000 threshold and learned that the penalty will be stopped. As a result, in the test episodes, the agent may have been able to leverage that learning to achieve stronger performance.

In the context of [3], we analogize this incentive structure to when the manager assigns a task for the day and allows the employee to do as they wish for the remainder of their time. As a result, we can reasonably expect a variety of behaviors such as completing the work and getting a leg up on future tasks or completing the task and then idling.

6.7 Episodic Rewards

As expected, we did not note significant performance increase with this incentive structure. As reasoned above, episodic reward relegates feedback signals to the end of an episode, and as such, gives the learning agent less information with which to update its performance. In the context of [3], we consider this mechanism analagous to the manager who only provides feedback once a year – at the annual review. As such, the employee does not get feedback on how their immediate performance
Across the board, we see the performance of episodic reward in line with the performance of a random agent. We note slight performance increase in "Episodic Rewards" and "Episodic Rewards Time Incentive."

7 Conclusions and Outlook

Results show that with different incentive structures, a more efficient mineral collection can be achieved. Interestingly, different incentive structures seem to work differently well for the implemented policy gradient methods. While episodic rewards lead to a similar performance of PPO and A2C, only time incentive and mineral thresholding lead to a stronger performance of A2C. As decisions seem to take longer for DQN, this method is outperformed by the other two for all investigated incentive schemes.
On a broader scale, it is possible to influence the behaviour of the agents differently with different incentive structures. However, due to the restrictions in time and calculation power, the training period within this project was not sufficiently high to make secure assumptions. Also, the multi-agent regime could not be investigated within the scope of this project. For future research, we therefore recommend longer training periods and the inclusion of multi-agent regimes. If it is possible to mimic the learning behaviour of humans by reinforcement learning, these models can be used to evaluate different incentives and policies before their implementation and make assumptions on the effectiveness of these before applying them to the real world.

8 References

[1] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. 2016, pp. 1928–1937.

[2] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[3] Charles T Munger. *Poor Charlie’s Almanack: The Wit and Wisdom of Charles T. Munger*. Donning Company, 2006.

[4] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[5] Oriol Vinyals et al. *StarCraft II: A New Challenge for Reinforcement Learning*. 2017.

[6] Wikipedia. *Q-Learning*. URL: `https://en.wikipedia.org/wiki/Q-learning`. (accessed: 26.11.2020).

[7] Wikipedia. *Reinforcement Learning*. URL: `https://en.wikipedia.org/wiki/Reinforcement_learning`. (accessed: 24.11.2020).

A Project Management

A.1 Time-Management

To successfully reach our goals we designed a timeline. In the beginning of the project it was important to gather a lot of information on the subject and read up on Reinforcement Learning. Also a set of milestones had to be designed to keep the focus on the right path (see Figure 8). After some research a suitable environment was found and the first attempts could be made. On the 05th of November we had a first successful simulation. Due to the high computational costs a request for the Euler Cluster at ETH was made to speed up the simulations.

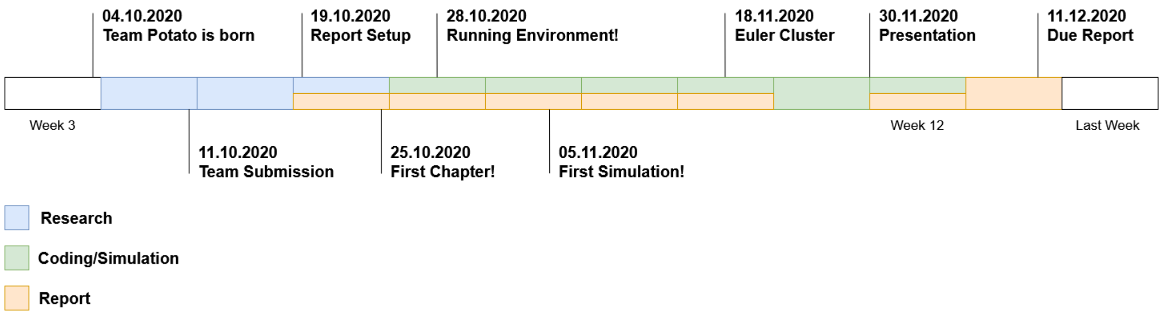


Figure 8: Timeline

A.2 Individual contributions

In the beginning we had a team of five people. Unfortunately two left the team potato so there were only three left:
Anna (BsC Electrotechnical) - responsible for report (structure and documenting the process)
Anya (PhD Electrotechnical) - responsible for presentation and Euler cluster and evaluated the results
Bryan(What do you do??) - the potato king, the one with the idea and the code master
A lot of task we worked on together and all the decisions were made in the whole team so there is no clear division of labor.

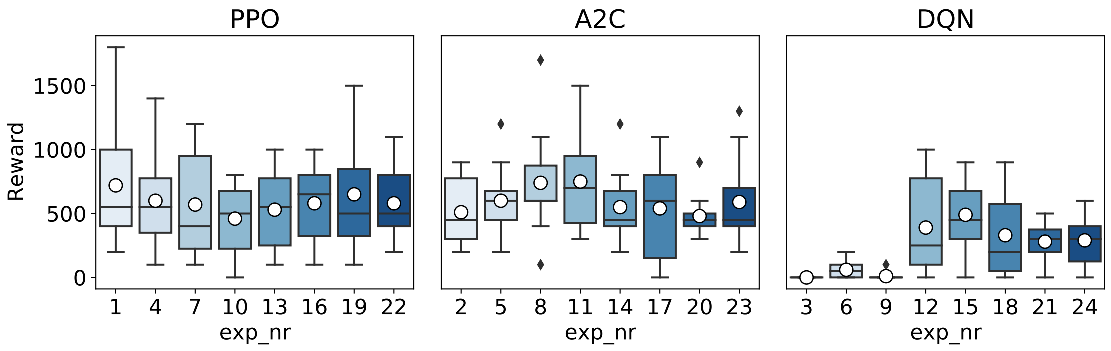
B Supplementary Results

For the first evaluations, we ran the models ten times and gathered the collected minerals. The results can be seen in table 1. It can be seen that some combinations of algorithm and incentives have better results while others lead to models that show worse results. The best results are obtained by experiments with numbers 1, 8 and 11, which correspond to PPO with no further incentive, A2C with episodic and the combination of time and episodic incentives.

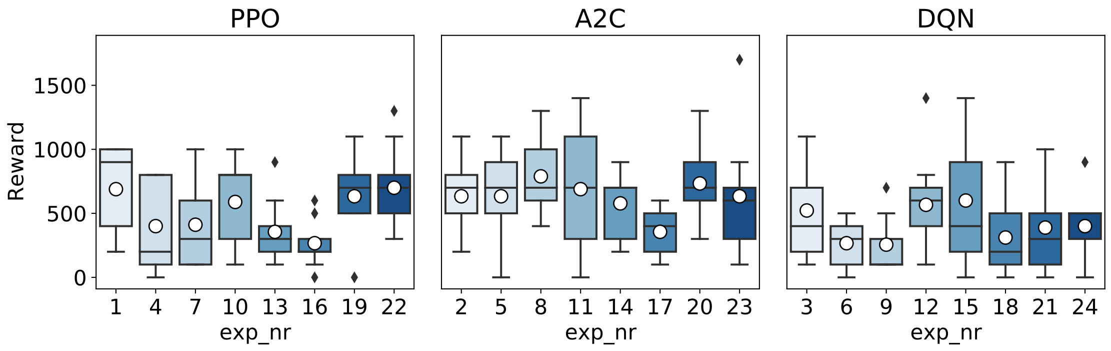
Table 1: Configuration and results for all the experiments.

exp_nr	Algorithm	Incentive	Min.	Max.	Mean	Median
0	random	none	0	1200	522	500
1	PPO	none	200	1800	720	550
2	A2C	none	200	900	510	450
3	DQN	none	0	0	0	0
4	PPO	time	100	1400	600	550
5	A2C	time	200	1200	600	600
6	DQN	time	0	200	60	50
7	PPO	episodic	100	1200	570	400
8	A2C	episodic	100	1700	740	600
9	DQN	episodic	0	100	10	0
10	PPO	episodic, time	0	800	460	500
11	A2C	episodic, time	300	1500	750	700
12	DQN	episodic, time	0	1000	390	250
13	PPO	thresholding	100	1000	530	550
14	A2C	thresholding	200	1200	550	450
15	DQN	thresholding	0	900	490	450
16	PPO	time, thresholding	100	1000	580	650
17	A2C	time, thresholding	0	1100	540	600
18	DQN	time, thresholding	0	900	330	200
19	PPO	episodic, thresholding	100	1500	650	500
20	A2C	episodic, thresholding	300	900	480	450
21	DQN	episodic, thresholding	0	500	280	300
22	PPO	time, episodic, thresholding	200	1100	580	500
23	A2C	time, episodic, thresholding	200	1300	590	450
24	DQN	time, episodic, thresholding	0	600	290	300

In a second run, the tests were performed eight times for every models in order to obtain the values displayed in figure 6. To evaluate the stability of our results, these two experiment runs are compared to each other to see how high the variation



(a) Results obtained by the first ten runs of experiments.



(b) Results obtained by eight runs used for the visualisation of mineral collection over time.

Figure 9: (For some of the experiments, significant differences can be observed, However, the overall distribution of collected minerals for the different configurations stays similar.

between two different testing sets is. Figures 9a and 9b show the results respectively. It can be seen that for single configurations the results differ quite significantly (e.g. exp_nr = 3, 6, 9, 13, 16). However, the more general shape and features seem to stay the same. DQN shows poorer performance in both cases and A2C the highest values. Experiments 1, 8 and 11 show the best values in both cases.