CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

AI Safe Exploration: Reinforced learning with a blocker in unsafe environments

Bachelor of Science Thesis in Software Engineering and Management

Marco Koivisto
Philip Crockett
Axel Spångberg

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

AI Safe Exploration: Reinforced learning with a blocker in unsafe environments

Marco Koivisto,
Philip Crockett,
Axel Spångberg

Supervisor: Piergiuseppe Mallozzi
Examiner: Rogardt Heldal

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

AI Safe Exploration: Reinforced learning with a blocker in unsafe environments

Philip Crockett
Department of
Computer Science and Engineering
Gothenburg University — Chalmers
Gothenburg, Sweden
guscrocph@student.gu.se

Marco Koivisto
Department of
Computer Science and Engineering
Gothenburg University — Chalmers
Gothenburg, Sweden
guskoima@student.gu.se

Axel Spångberg
Department of
Computer Science and Engineering
Gothenburg University — Chalmers
Gothenburg, Sweden
gusspaax@student.gu.se

***Abstract*—Artificial intelligence can be trained with a trial and error based approach. In an environment where a catastrophe can not be accepted a human overseer can be used, but this might lower the efficiency of the learning. The study includes implementation of an artifact meant to replace the human overseer when training an AI in simulated unsafe environments. The results of testing the implemented blocker shows that it can be used for avoiding catastrophes and finding a path to reach the goal in 17 out of 18 runs. The single failed execution shows that the implemented blocker is in need of improvement in terms of data efficiency. Shaping rewards solely to reduce number of steps and catastrophes for a reinforcement learning agent has been done successfully to some degree, but further steps can be taken to lower the number of catastrophes and steps.**

***Keywords*-Artificial intelligence; Reinforcement learning; Safe exploration; Blocker; Machine learning; Baby AI Game; Gym Mini Grid;**

ABBREVIATIONS

AI	Artificial Intelligence
RL	Reinforcement Learning

I. INTRODUCTION

Artificial Intelligence lets a computer perform certain tasks without the help of a human, in some cases AI is used for tasks which humans are not able to solve. But there are also tasks where AI is used to optimize tasks performed by humans. Building an AI today can be done using neural network data structures, consisting of actions and rewards given to a Reinforcement Learning agent, with the purpose of teaching a RL agent in terms of actions and rewards for taking a certain action [1].

A. Motivation

An increasing number of industries are using AI, examples includes the car and vacuum cleaner industries. To train an AI to perform according to specification, RL is used. RL is a learning process for AI where it acts on trial and error and learns from mistakes or successful actions that are made. RL is not suitable for environments where a trial and error approach is unsafe for either the environment or the agent [2]. A scenario with a robot vacuum cleaner where it during its training phase breaks a vase, it learns from the performed action, but the vase is already broken. Similar scenarios can be seen within the car industry, but with more catastrophic consequences. As motivated by W. Saunders, G. Sastry, A. Stuhlmüller and O. Evans these kinds of systems can potentially be dangerous, since an AI has not learned to avoid catastrophic actions from the beginning [2].

The authors solve the problem of training an AI with reinforcement learning in a safety critical environment by adding a human overseer [2]. Meaning that a human would monitor the behavior of the agent at all times. Before the agent is about to act in a way which does not meet the requirements or cause a catastrophe, it is stopped by the human and is given a negative reward. In that way the agent learns what bad behavior is, without ever finishing the action. W. Saunders, G. Sastry, A. Stuhlmüller and O. Evans compares this learning process to when a licensed driver oversees and teaches an unlicensed driver [2].

One problem with human intervention is that it lowers the efficiency of the learning process. Since it means that a human needs to monitor the behaviour of the AI during training [2]. Computers can analyze whether proposed actions by an agent are correct or not, faster than humans. However human intuition can currently not be replicated on a computer. Where can a balance between the advantages of using a human or a computer be found in terms of training time for a RL agent, in order to optimize the learning while not risking the safety of a RL agent or an environment?

B. Background

The study was inspired by research previously done by W. Saunders, G. Sastry, A. Stuhlmüller. The authors of the paper evaluates reinforcement learning in unsafe environments using human intervention during the RL agents training process [2].

C. Contributions

This study includes an implementation of a ”blocker” inside an AI simulated environment which has not been done earlier. The blocker works by specifying unsafe actions in an environment, to protect the safety of the agent and the environment. Using the vacuum cleaner example, a requirement specifying not to break vases could be added. In a perfect software

environment where the results of checking whether the next action is to break a vase always will be correct, compared to a real world environment. The requirement based blocker is implemented with the purpose of letting an RL agent learn without acting in a way that breaches the requirements which are known. The blocker lets an RL agent learn in an unsafe environment without using a human overseer during training, unlike the research done by W. Saunders, G. Sastry, A. Stuhlmüller. This study will evaluate different configurations of the blocker, by giving the RL agent different rewards, to measure steps taken and number of blocked actions which prevent catastrophic actions in relation to rewards given.

II. Research Methodology

This study was executed using design science. The research method was chosen since as explained by the authors A. Hevner, S. March, J. Park and S. Ram it should be used when the research includes creation of a new artifact which can solve a practical problem [3].

A. Problem identification and motivation

The practical problem which was addressed in this research is related to reinforcement learning for AI. Human intervention is one approach when training an agent with reinforcement learning in a safety critical environment, in order to not cause harm to itself or the environment [2]. The solution in this research offers an alternate approach to human intervention when training an RL agent in an unsafe environment.

B. Research Questions

1) To what extent could reinforcement learning in a safety critical environment be performed with a ”blocker” extension?
2) To what extent could rewards be shaped for the reinforcement learning in order to minimize the number of blocked catastrophes?

The first research question was selected since the human intervention approach by itself can be too ineffective [4]. The study will investigate whether reinforcement learning in an unsafe environment can be performed with a blocker. The second research question was chosen in order to evaluate and optimizing the blocker by researching if different rewards can be used to minimize the number of prevented catastrophes.

C. Objective for solution

The study includes implementation of a blocker inside the AI simulation environment ”Baby AI Game” [5]. The purpose of the blocker is to let an AI learn in a safe manner, meaning not entering unsafe states. Unsafe states in this study is defined as water which destroys the agent.

The environment can have states which are unsafe, but with the use of the implemented blocker it should still be able to learn with trial and error. The blocker gives an option to add the known states which are unsafe for the agent. When the agent is training, the blocker checks whether the next suggested action is breaching the specified rules of the blocker. If the proposed action is considered unsafe based upon the rules, the agent will receive a negative reward and get a new action which is safe, which is preventing a catastrophe from occurring. In this implementation the agent is given a ”wait” command as a safe action when the agent is blocked from entering an unsafe state.

D. Description of Artifact

The blocker artifact is developed as an extension to the open source project ”Baby AI Game” [5] and the training of the agent is performed in the open source environment ”Minimalistic grid world environment” [6]. The artifact also includes an unsafe environment for testing purposes. A water tile has been constructed in the ”gym-minigrid” [6] project, the water is defined as unsafe in the blocker. The artifact also includes a tool that pseudo randomly generates an unsafe environment, meaning that the water tiles are positioned at a random location inside the environment. Using a specific random seed, so all environments can be recreated again. An evaluator has also been implemented which is used to save the results from training, in relation to the evaluation variables discussed later in this section.

The work uses a fork created by Piergiuseppe Mallozzi [6], [5]. The original repository is created by Maxime Chevalier-Boisvert [7] and [8]. Below are links to Piergiuseppe Mallozzi repositories and the created implementations of this study is located at the branch ”random_envs”.

1) Extensions to ”Baby AI Game”
2) Extensions to ”Minimalistic grid world environment”

E. Presentation of Artifact

The figure 1 describes an abstract explanation of how the blocker acts. The reinforcement learning agent suggests an action to perform, the blocker then checks whether the action suggested is safe based upon the rules specified in the blocker. When the action is evaluated as safe, the action and reward is not altered, but if the action is evaluated as unsafe a ”wait” action is given by the blocker and a negative reward is given to the agent for proposing the unsafe action.

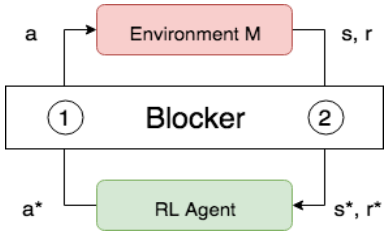

Fig. 1: Process of the blocker

See figure 1 the RL agent proposes an action a^*. If the action violates the blocker rules, the blocker alters the action to be a safe action a in the environment. At (2) the environment returns the result of the step s and a reward r, the blocker then alters the reward the RL agent is given, if it violates the blocker rules the reward $r*$ is the altered reward of r. The

environment gives a reward r if the agent reaches the goal and the reward is unaltered by the blocker.

F. Implementation of Artifact

The implementation of the artifact is located in both the [6] and [5] repositories. The relevant files are *envelopes.py* [6], *absence.py* [6], *env_generator.py* [5], *evaluator.py* [5] and *extendedminigrid.py* [6].

envelopes.py
: Contains the environment wrapper that captures each step which the agent performs in the environment. This effectively acts as our blocker implementation, each step suggested is investigated to decide if the action is unsafe or if the action is safe and changes the rewards given to the agent if a catastrophe occurred, or if it died.

extendedminigrid.py
: Is an extension of the environment class "MiniGridEnv" found in [6]. It also contains the environment observation code that is used to decide the type of tile in a direction from the agent, e.g. left, right or in front.

absence.py
: This class sets up a state space for the blocker, preventing the agent from entering states that are of type "violation" in the configuration. For example, in this artifact, water was defined as a violation.

evaluator.py
: Saves the results gathered during training of the RL agent. The results are saved in form of .csv files.

env_generator.py
: Creates pseudo random environments and their respective configuration files. This is used to evaluate if the blocker implementation differentiate from runs without a blocker. The pseudo random environments can be recreated using the same seed id. Thus all pseudo random environments can be recreated, in case a certain environments presents "interesting" data. It provides a solution for recreating the same environment for further investigation.

The dependent variables needed for evaluation is defined in a JSON configuration file. The reward can be altered from a separate configuration file, which enables easy change of configuration for training the agent. What rewards that work best in terms of how quick the RL agent learns the path to the goal, can have a correlation to different rewards. This makes the reward parameter important to easily be changed.

G. Description of rewards

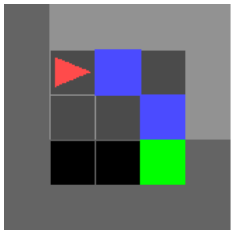

Fig. 2: Unsafe environment in a size of 5x5 tiles - The blue tiles in the picture represents water.

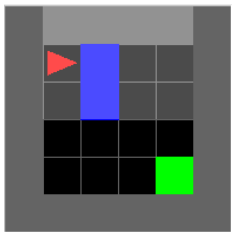

Fig. 3: Unsafe environment in a size of 6x6 tiles - The blue tiles in the picture represents water.

Examples of unsafe environments used in this study can be seen in figure 2 and figure 3. The example environments shown contains two water tiles each, which in the evaluation is randomly placed. The goal of the agent is to navigate to the green square without ending up on a water tile. When the agent reaches the goal, it is rewarded with 1000 points and for each step it takes (including turns) it is given -1 point. Meaning that the final reward when it reached the goal is $goal\_reward - steps$. In figure 3 the maximum reward or the most efficient way to the goal is when the RL agent enters the goal with a final reward of 993 points. Since the least number of steps taken to reach the goal is 7 steps. In the evaluation the learning stops when the agent enters the goal with a total mean reward greater than 985. The final reward has been chosen since it gives room for extra steps needed dependent on how the water tiles are placed. Thus the same final reward is used for the different configurations which are compared to each other and are evaluated with the same conditions.

The reward shaping means that the minus reward is altered when the agent proposes to perform an unsafe action which is blocked. The three different minus rewards used in this study is -1, -100 and -1000. Negative rewards between -1 and -1000 enables to investigate if higher negative rewards cause fewer catastrophes and how the different rewards affect the number of catastrophes. This is done in order to answer the second research question.

Since the blocker changes the reward the agent receives when blocking catastrophes and when it dies, it affects the cumulative reward the agent receives when reaching the goal. Meaning that the total reward of one iteration where the goal was reached, results in the reward for reaching the goal

minus the number of negative rewards given by proposing catastrophic actions. The goal_rewards variable in this study is set to 1000. Where each proposed catastrophic action has a x minus reward. The used variables is: ”violation”, ”death”, ”step” and ”goal”. The rest of the variables inside the configuration file is set to zero.

In the artifact the maximum number of steps the agent is allowed to take is defined as following $4 * grid\_size * grid\_size$ where $grid\_size$ is the width of the grid, e.g. if the grid is 6x6 the $grid\_size = 6$. Meaning that in a 5x5 environment the maximum number of steps is $100 = 4 * 5 * 5$ and in 6x6 it is $144 = 4 * 6 * 6$. When more steps than the limit is taken the agent starts over from the beginning. The agent is always positioned in the top left corner upon death and when it runs out of steps. The agent only receives a negative reward for each step it has taken when it reaches the goal, this is what keeps the agent searching for a shorter or better path to its goal. The goal is always positioned in the bottom right corner of the environment.

H. A2C Algorithm

The reinforcement learning algorithm used in this experiment is an Actor-Critic-Advantage algorithm (A2C) which is a synchronous version of the (A3C) algorithm. A2C is a deep learning algorithm which creates a state map, where each state is mapped with a given reward. ”The goal of the agent is to maximize the expected return from each state” [9]. Thus the agent learns to avoid negative rewards and seek out the positive ones, it can perform a negative action if the final reward is still greater. Which makes a blocker more important, if some states should not be entered. The algorithm learns during its different periods which action had the highest reward, but the algorithm also takes in consideration that the learned and safe actions may not be the most optimal route in terms of rewards. The algorithm uses the variable ”entropy” to discover new possible paths were the goal is to eliminate the possibility that the agent learned a sub-optimal path. The agent sometimes finds itself in a scenario were all the actions the agent attempt will lead to a negative reward, the algorithm then uses the variable ”policy” to decide which actions and paths gives the lowest minus reward thus making the agent always look for the most beneficial path.

I. Evaluation Method

Evaluation of the blocker is done by self defined key metrics which are related to the results of learning by the agent. Each configuration is executed using 48 processes to reduce the time to train the agent. The configurations are executed on different docker containers with similar hardware specifications, so runs can be executed in parallel.

The 3 different reward configurations of the agent will each be executed in the two different environment sizes seen in figure 2 and figure 3. Each execution of the rewards will be executed 3 times in each environment size where the water has randomly been placed. The mean of the 3 executions will then be used for comparison between the different configurations. The described executions will be done both with the use of the blocker and without the blocker. Which gives results of 3 different rewards in 2 different environment sizes, with and without the blocker. These results with and without the blocker will then be compared to each other to answer RQ1 and RQ2. Each of the evaluations is stopped when the same total reward mean has been achieved to ensure that the results can be compared to each other. This is done by as earlier described, stopping the training when a final reward of greater than 985 is achieved. The run is then exited and the results are saved to a comma separated value (csv) file. The file contains the data of the evaluation variables discussed below.

J. Evaluation variables

The evaluation variables are results of when the agent has been trained until the goal has been found with a final mean reward greater than 985.

1) Number of steps: The number of steps the agent takes. Every step the agent performs including, left turn, right turn, wait or stepping forward is counted as a single step.

2) Number of deaths: The number of deaths is a count of when the agent steps into water.

3) Number of blocks: The number of blocked catastrophes. This variable is used to measure how many blocked actions the agent gets.

K. Evaluation of research questions

1) RQ1: Is answered by comparing the executions in the different environments with the implemented blocker to the executions without the blocker. This done to show that the sub-shortest path can be found with the use of the blocker.

2) RQ2: Is answered by comparing the different executions with the different rewards towards each other and then see if any patterns can be found in the evaluation variables, number of catastrophes and number of steps.

L. Validity Threats

The validity threats presented below has been identified by the use of the template that C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell and A. Wesslén discusses [10].

1) Conclusion validity: The actions of the agent are randomly taken until the agent starts learning from the rewards. The random steps in the start can have an impact on the results. This risk was reduced by running 3 executions with the same reward, each in the same size of the environment but where the water tiles are randomly placed. The mean of the results will then be used to compare the different configurations.

2) Construct validity: The collection of results from training the reinforcement learning agent using different blocking methods needs to be reliable and validated. Thus the collecting of data takes a long time. This threat is avoided by pretesting the evaluation of the agent in smaller time periods before the concluding tests are executed.

3) Generalizability: The developed blocker artifact is based upon previous work done by M. Chevalier-Boisvert and P. Mallozzi, the artifact implemented will only work generically in the environments used in this report. Meaning the different blocker methods used only can be replicated inside of the Baby-AI-Game and the environment inherited from Gym-minigrid. The risk is accepted since a more general blocker implementation which works in other simulation environments is not feasible because of time constraints.

III. Results

The collected data has been compiled into tables and diagrams. All plots have been created using "matplotlib" in python. The tables show data from each of the individual executions in terms of reward configuration and environment setup. The full set of data can be found in form of comma-separated value (csv) files here. Each figure represents an execution using 48 processes. The "Blocks" in the tables shows the total number of blocked catastrophes during training, a block is counted when the agent proposes to step into water. The "Deaths" is the number of times the agent gets destroyed during the training and "Steps" is the number of steps until the agent learned a path to the goal with a final reward of 985 or above. The mean values are rounded to the closest whole number. The seed id in the tables can be used for looking at the environment which the training was executed in. Visualizations of the environments is found here.

In the presented figures 4 and 6 the N_updates in the x-axis is the number of updates, 1 update is roughly 240 steps, e.g. the number of updates times 240 yield an estimate of the total amount of steps taken in a run. In the figures 4 and 6 the presented lines are N_deaths or N_blocked_actions, N_step_per_episode and N_goal_reached. The figures highlight the convergence of finding the goal for the agent. The number of steps per episode shows the average steps taken by the 48 processes as it gets lower, it starts finding the goal more reliably. The convergence between N_step_per_episode and N_goal_reached highlights that the agent is done with the training. When they converge the training is over. Graphs representing every execution can be found here.

TABLE I: Environment 5x5 -1 reward

With blocker

Seed id	Blocks	Deaths	Steps
29f901e5	1801	0	3 629 040
9fec1e45	536	0	326 640
740af178	93	0	1 615 440
Mean	810	0	1 857 040

Without blocker

Seed id	Deaths	Steps
29f901e5	995	10 226 640
9fec1e45	3965	96 240
740af178	603	5 469 840
Mean	1854	5 264 240

TABLE II: Environment 6x6 -1 reward

With blocker

Seed id	Blocks	Deaths	Steps
2a6bd7a3	884	0	177 840
2122c6c6	108	0	127 440
f3c8304e	2939	0	353 040
Mean	1310	0	219 440

Without blocker

Seed id	Deaths	Steps
2a6bd7a3	147	84 240
2122c6c6	113	115 440
f3c8304e	699	336 240
Mean	320	178 640

In table I all the runs and their respective seeds that were used to create their environment is gathered. The table shows the different runs in the respective environments. Note that a seed id is used for both the blocker and the execution without a blocker, so the results which are compared have been tested inside the same generated environment.

TABLE III: Environment 5x5 -100 reward

With blocker

Seed id	Blocks	Deaths	Steps
3030dae8	64	0	343 440
c5484d3c	581	0	984 240
ceaf9555	738	0	3 893 040
Mean	461	0	1 740 240

Without blocker

Seed id	Deaths	Steps
3030dae8	44	741 840
c5484d3c	181	24 240
ceaf9555	279	14 997 840
Mean	168	5 254 640

TABLE IV: Environment 6x6 -100 reward

With blocker

Seed id	Blocks	Deaths	Steps
0ff854c7	8408	0	2 193 840
6c473d49	22	0	305 040
bc1952c7	727	0	1 680 240
Mean	3053	0	1 393 040

Without blocker

Seed id	Deaths	Steps
0ff854c7	20115	1 382 640
6c473d49	33	540 240
bc1952c7	89	804 240
Mean	6746	909 040

TABLE V: Environment 5x5 -1000 reward

With blocker

Seed id	Blocks	Deaths	Steps
0a3fed73	680	0	2 647 440
52b1968b	327	0	785 040
62d6a9a4	78	0	547 440
Mean	361	0	1 326 640

Without blocker

Seed id	Deaths	Steps
0a3fed73	202	3 350 640
52b1968b	29	1 097 040
62d6a9a4	79	482 640
Mean	103	1 643 440

TABLE VI: Environment 6x6 -1000 reward

With blocker

Seed id	Blocks	Deaths	Steps
005cf200	483	0	941 040
9b03d45f	131	0	300 240
fcc075cb	56	0	69 840
Mean	223	0	437 040

Without blocker

Seed id	Deaths	Steps
005cf200	127	458 640
9b03d45f	10	96 240
fcc075cb	39	374 640
Mean	59	309 840

Tables III, V, IV and VI presents the data where higher negative rewards were given to the agent when it died or was blocking catastrophes.

A. *Uncompleted configurations*

The runs in figures 4 and 6 never finished to successfully learn the path to the goal with a final reward of 985 or above during the training. They did not complete because their processes died, the blocker configuration ran for 3 hours of training and without blocker died after 10 hours of training. In the runs with a blocker, we noticed that memory usage was high which caused the processing speed to slow down as the training continued. The configuration without a blocker never grew noticeably in memory usage, instead we cannot pinpoint as to why the execution terminated. It should be noted that it took the most steps of all runs without reaching the goal, whilst the blocker configuration in the same environment reached the goal as seen in figure 7. However the graph for figure 6 shows that it was close to converging, but afterwards started diverging again.

IV. Discussion

A. *RQ1 - To what extent could reinforcement learning in a safety critical environment be performed with a "blocker" extension*

As presented in the results all the executions using blockers except one has been completed successfully, meaning that they have learned the path to the goal with a reward of 985 or above without dying. The single execution with seed id $740af178$ in table I and figure 4 had 1 615 440 steps, the execution was

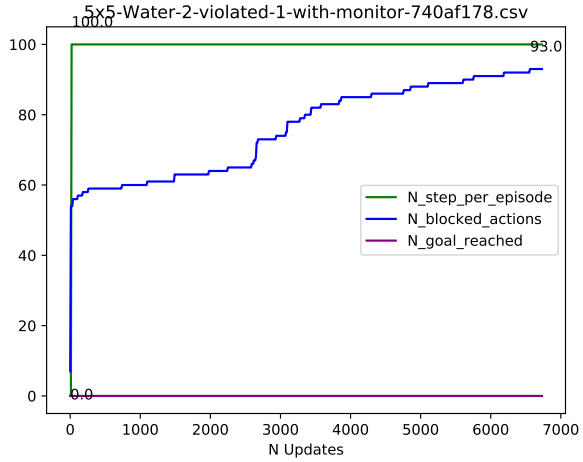

Fig. 4: Uncompleted 740af178 with blocker 5x5 -1

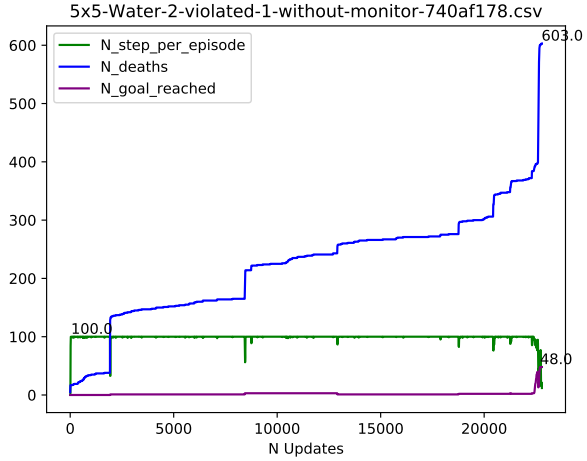

Fig. 5: Completed 740af178 without blocker 5x5 -1

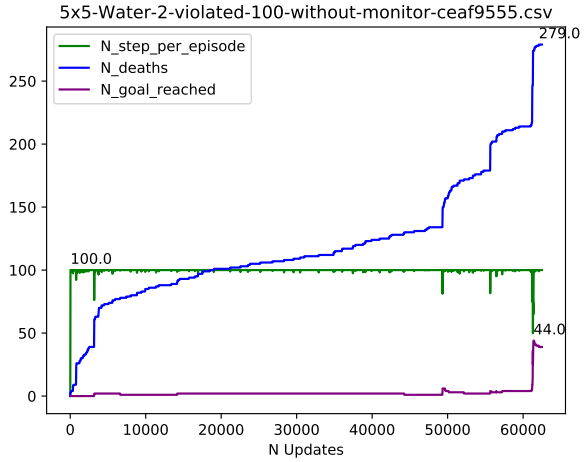

Fig. 6: Uncompleted ceaf9555 with blocker 5x5 -100

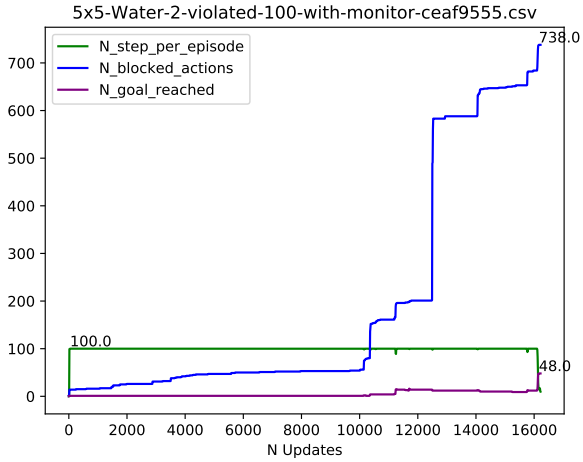

Fig. 7: Completed ceaf9555 with blocker 5x5 -100

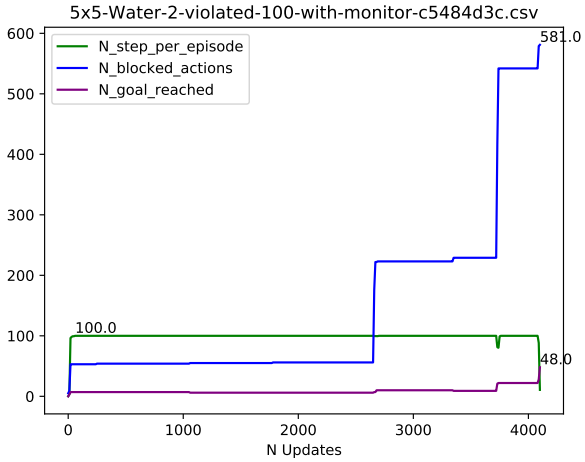

Fig. 8: 5x5 reward -100

terminated since the process used all available RAM memory on the computer. This could be due to the state space used to monitor the agents actions and ensure it does not enter the water, grows in terms of memory usage. After 1 hour of running it was using nearly 2GB of memory, compared to an execution without a blocker where the memory usage never went above 310MB. Meaning that with a more data efficient blocker implementation more time-consuming training might have been able to finish. The second execution which was not able to finish was seed id $ceaf9555$ in table III and figure 6. This execution stopped with an amount of 14 997 840 steps, meaning that the data efficiency of the blocker implementation might not have been the only problem.

B. RQ2 - To what extent could rewards be shaped for the reinforcement learning in order to minimize the number of blocked catastrophes?

Our hypothesis for the blocker runs was that a higher negative reward would result in fewer catastrophes at the cost of more steps needed to find the path to the goal with a final reward of 985. This can be seen in table III and table V where in the -1000 reward the mean value for number of steps and blocks is lower for higher negative rewards.

Our findings agree with W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evan, thus they discuss using a high negative reward where causing catastrophes to reach the goal is infeasible, learning the agent to avoid catastrophes [2]. Whilst they were interested in investigating about learning an agent when it causes a catastrophe our focus was to research avoiding catastrophes all together, with usage of a blocker. As mentioned by them, reinforcement learning is insufficient to achieve safety because it learns by trial and error [2]. Meaning the agent using a blocker will avoid causing catastrophic events but still receives the negative reward as if it had caused the event. An example of this is presented in figure 8 and figure 9.

Different configurations do perform differently, but we cannot say that the configurations alone are the reason for the

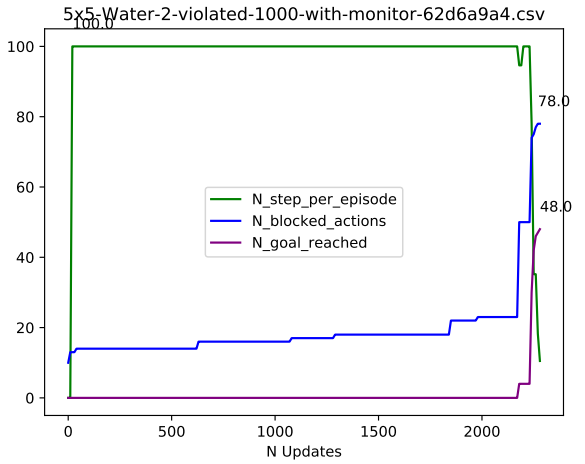

Fig. 9: 5x5 reward -100

different results. The initial thought was that the environment should not matter in terms of how many catastrophes the agent performed, that only the negative reward would teach it to not do the same action again, no matter where the catastrophe was located. As discussed in W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evan the catastrophes negative reward should be much lower than the reward for reaching the goal, where the agent learns to cause a catastrophe to reach the goal should never be worth it [2]. However, the findings shows that it is not simple enough to tune the rewards to a higher negative reward and expect fewer catastrophes. This can be seen in figure 10 where the agent died roughly 20000 times, which is higher than all the other runs. The same environment seed in a configuration with a blocker also blocked the most actions at roughly 8000 times, the run is shown in figure 11 This environment is seen as an outlier in the collected data, since it differs wildly from the rest of the data. This shows that there are environments where there can be many catastrophes

even with a higher negative reward, in relation to how ”easy” it is to cause a catastrophe, the environment where this data was seen is shown in figure 12. We cannot say with accuracy that the environment is the sole cause of these catastrophes. Further investigation is needed to surely conclude this.

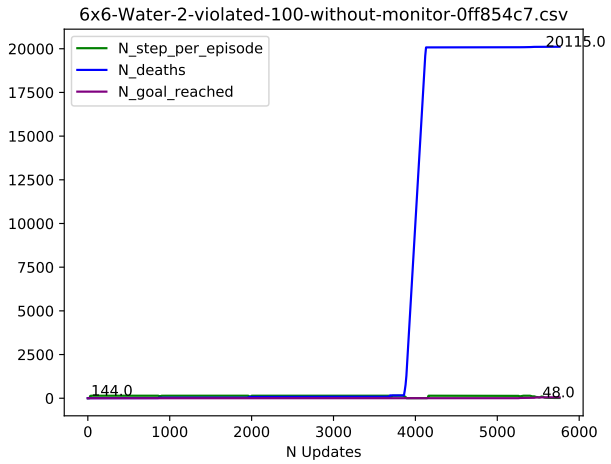

Fig. 10: 6x6 -100 without blocker run

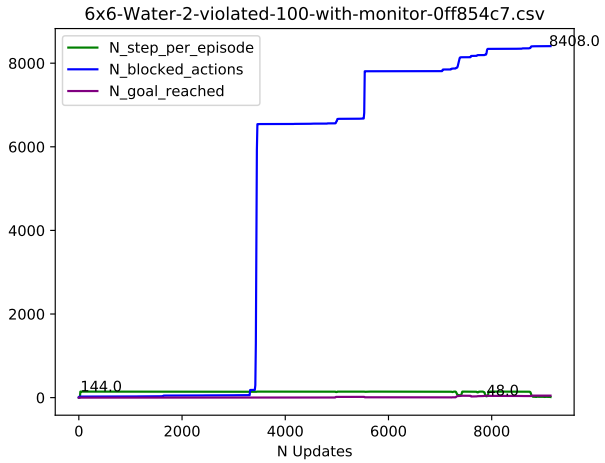

Fig. 11: 6x6 -100 blocker run

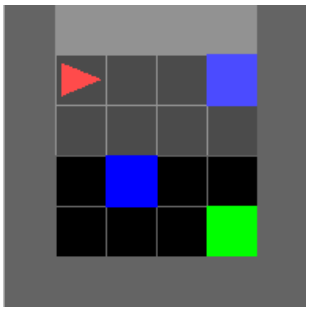

Fig. 12: Environment with outlier data

Running without a blocker with a higher negative reward lowers the number of deaths in some runs. Generally for the data it can be seen that running without a blocker converges faster to the goal as seen in for example the table V. The mean value for number of total steps it takes is lower, but this is likely because it restarts from the beginning when dying whilst the blocker continues running after being blocked. Meaning that it allows the agent with the blocker to always perform the max number of steps in the environment.

Early in training the agent causes some deaths or blocked actions and learns from the mistakes. As the agent starts to converge the number of deaths or blocked actions starts to increase, this may be because the A2C agent tries to find a path where it can cause a catastrophe and still receive a higher final reward. Meaning that it is verifying that the path to the goal which has been found is the most reward intensive one. Potentially, in figure 6 we see that it starts causing an increasing number of deaths at as it nears its convergence, but it however starts to diverge before it reaches a total reward mean of 985 or above. If it had continued running it is possible that the converging would have started again.

C. Related work

As explained by D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman and D. Mané using different polices for exploration in reinforcement learning could improve the agents ability to avoid catastrophes [4]. But choosing a more sophisticated policy can also cause more harm. Since the policy can be worse than random actions. In the results the agent has been using a random exploration policy, no modifications have been done to the policy of the agent. As such, the results shows that a random exploration policy works for learning.

According to D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman and D. Mané using hard coded catastrophe avoidance by the use of an blocker works best in an understood domain with few tasks where catastrophes have been identified in beforehand [4]. However, in real life situations it becomes increasingly complex to classify a blocker with catastrophe prevention. Because unknown catastrophes will still happen, if the blocker does not know that it should block it from happening.

In ”Concrete problems in AI safety” [4] and ”Trial without Error: Towards Safe Reinforcement Learning via Human Intervention” [2] they both reach the same conclusion about human intervention in reinforcement learning reaches the problem of scalable oversight, where the agent requires too much exploration in order for the human oversight to be practical. In our case, there is no human intervention required for the catastrophes to be blocked, instead we assume that the different catastrophes are known before the training.

The challenge with a blocker is ensuring that the model is not miss-specified [4]. We agree with this, as a problem of doing simulated training with requirements, is that the trained agent only operates within the bounds of the specified requirements. If another kind of catastrophe can occur, the

agent would not have an idea about that it was a catastrophe and possibly cause harm. Modeling perfect requirements before having any real world data is near impossible to do, no matter how advanced a simulation is, a real world setting will have new kinds of interactions and unknown catastrophes will emerge. We think that a blocker is only the beginning, further research is required to reach a good sentiment. It works best in well-defined domains, with a little real life interaction, meaning for example a robot in a factory without humans and predefined working areas. But in a setting with ”floating” variables, such as humans walking around the robot, which was not present during training, is an ill-suited area for using a blocker to prevent catastrophic events. [4].

V. Conclusion

A. RQ1: To what extent could reinforcement learning in a safety critical environment be performed with a ”blocker” extension?

The results show that a reinforcement learning agent can successfully learn the path to the goal with a blocker, also meaning without causing any catastrophes, in the same setting as the none blocker executions in 17 of 18 executions. The single execution which was not able to finish can have a relation to the blocker being too data inefficient, since all memory was used and the training stopped. This can not be fully concluded since all resources in terms of memory was used by the blocker, the blocker would need to be more data efficient to gather more accurate results for longer training periods. The results show an indication on the blocker being a working alternative for human intervention but more tests with statistical analysis is needed to conclude this.

B. RQ2: To what extent could rewards be shaped for the reinforcement learning in order to minimize the number of blocked catastrophes?

The results show that to some extent using different rewards decreases the number of steps and catastrophes in our executions. However, there are cases where this is not true, see figure 11 and figure 10 where a run in a certain environment caused more deaths, blocked more actions and took more steps than any other environment. However the different configurations in the environments shows a trend that higher negative rewards causes fewer blocked actions and deaths. Further analysis of higher negative rewards could potentially highlight the validity of higher negative rewards for reinforcement learning trained agents.

It should be noted that shaping rewards is only one of the possible aspects of decreasing the number of blocked actions and deaths. In this study the A2C agent caused more deaths and blocked actions at the end of the executions. When the agent starts converging the goal, the agent verifies that the path is the most effective one in terms of reward. This is done by causing death and blocked actions to confirm there is no path which results in a higher reward, even though it includes negative rewards. Thus changing policy levels and exploration rate for the reinforcement learning algorithm could potentially decrease the number of deaths and blocked actions. But as the agent learns by trial and error, deaths and blocked actions are necessary in order for the agent to learn.

VI. Further research

A. Further development

The development of the artifact and the findings in this paper can be used in further research of a blocker extension with an RL agent.

1) Data efficiency: By increasing the data efficiency of the blocker implementation, longer runs would be able to run faster and not use all available memory on the computer it is running. By for example using more space efficient data types.

2) Foresee future moves: By checking more than one step in front of the agent, the blocker could see if it is heading towards a catastrophe in an earlier stage (e.g. in our case for example 2 tiles away). It could then give a lesser minus reward when the reinforcement learning agent is taking steps towards the catastrophe and a greater minus reward when the reinforcement learning agent tries to enter a water tile.

3) Propose a new safe action: Steering the agent away from the catastrophe. Instead of sending wait commands to the agent, the blocker could actively suggest a new route for the agent. With the purpose of reducing the number of steps it takes to find the goal.

4) Experiments with exploration policy's: Further experiments could include altering of the policy and entropy levels to alter the rewards and exploration in a more efficient manner. This can also be done by favoring reaching the goal instead of finding the most effective way. If the end goal is to learn safely, how important is it to use the most optimal route?

B. Combination of human intervention and blocker in a real world setting

When training in a real world environment looking for catastrophic events could be done with the use of sensors. When using sensors it cannot be assured that the retrieved results are never faulty. This creates a scenario where the blocker can not be fully trusted to decide whether the coming action is catastrophic or not. Further research of using a blocker when learning an agent could therefore instead of instantly stopping the agent from an action, notify a human to check whether the upcoming state is unsafe or not. In practice this could optimize the learning process where human intervention is used. Since the human would not have to monitor the robot at all times but only to give input when the robot is going towards a state which might be unsafe. As mentioned earlier the problem with human oversight is that the agent may need too many exploratory steps for a human to oversight. Only using a human when there is a possible catastrophe happening, could optimize this problem [4].

References

[1] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[2] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," *arXiv preprint arXiv:1707.05173*, 2017.
[3] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
[4] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565
[5] P. Mallozzi, "Baby ai game," https://github.com/pierg/baby-ai-game, 2018.
[6] ——, "Minimalistic gridworld environment for openai gym," https://github.com/pierg/gym-minigrid, 2018.
[7] M. Chevalier-Boisvert, "Baby ai game," https://github.com/maximecb/baby-ai-game, 2018.
[8] ——, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.
[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.
[10] C. Wohlin, A. von Mayrhauser, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, ser. International Series in Software Engineering. Springer US, 2012. [Online]. Available: https://books.google.se/books?id=3aPwBwAAQBAJ

APPENDIX

1) Extensions to "Baby AI Game".
2) Extensions to "Minimalistic grid world environment".
3) Execution results in graphs.
4) Execution results in comma separated files (csv).
5) Images of the environments.