APPLYING REINFORCEMENT LEARNING TO THE WEAPON ASSIGNMENT PROBLEM IN AIR DEFENCE

Hildegarde Mouton, Council for Scientific and Industrial Research, South Africa; Jan Roodt, StoneToStars Limited, New Zealand; Herman le Roux, Council for Scientific and Industrial Research, South Africa

Abstract

The modern battlefield is a fast-paced, information-rich environment, where discovery of intent, situation awareness and the rapid evolution of concepts of operation and doctrine are critical success factors. A combination of the techniques investigated and tested in this work, together with other techniques in Artificial Intelligence (AI) and modern computational techniques, may hold the key to relieving the burden of the decision-maker and aiding in better decision-making under pressure. The techniques investigated in this article were two methods from the machine-learning subfield of reinforcement learning (RL), namely a Monte Carlo (MC) control algorithm with exploring starts (MCES), and an off-policy temporal-difference (TD) learning-control algorithm, *Q*-learning. These techniques were applied to a simplified version of the weapon assignment (WA) problem in air defence. The MCES control algorithm yielded promising results when searching for an optimal shooting order. A greedy approach was taken in the *Q*-learning algorithm, but experimentation showed that the MCES-control algorithm still performed significantly better than the *Q*-learning algorithm, even though it was slower.

Introduction

In his 2007 book, *The Utility of Force – The Art of War in the Modern World*, General Rupert Smith concludes that modern warfare will call for information technology to support operations by helping the commander to understand the actions of the opponent and to separate him from the people.[1] Computational (software) agent systems are showing much promise in

Scientia Militaria, South African Journal of Military Studies,
Vol 39, Nr 2, 2011, pp. 1-15.

this regard, allowing researchers to understand complex human interaction and decision-making paradigms. One of the key design-parameters for agent systems is finding the appropriate learning algorithms.

One such a family of (machine) learning algorithms is RL. The latter is a branch of AI[2] and uses a formal framework defining the interaction between a learning agent and its environment in terms of states, actions and rewards. It is concerned with how an agent should take actions in an environment to maximise some sort of long-term reward.[3]

What sets RL apart from other machine-learning methods is the fact that the agent is not told which actions to take, but instead must discover by trial and error which actions yield the highest reward. All RL agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. The agent has to operate although it does not have exhaustive information or knowledge of the environment. This is particularly useful and important in applications such as WA where the necessary supervision and rules of engagement are known, but may be open to interpretation in the heat of battle.

The WA problem, stated very simplistically, is the assigning of *n* weapons to *m* targets.[4] As with the resource-allocation problem, the WA problem is also NP-complete and as such, no exact methods exist for the solution of even relatively small-sized problems. WA decisions are considered more easily quantifiable than threat evaluation (TE), and thus the challenge lies more in the solution methodologies of the problem rather than in the formulation, as is the case with TE.

In this article, we shall discuss whether RL was found to be suitable for solving the WA problem. A short overview of RL and related work is given, as well as an explanation where RL fits into the greater context. Command and control (C2), as well as threat evaluation and weapon assignment (TEWA), are discussed. Finally, the article explains how the WA problem was modelled and how the MCES-control algorithm and the TD-algorithm, *Q*-learning, were applied to the WA problem. The results are compared and we conclude with ideas for future study.

Overview of reinforcement learning and related work

In some of the latest research on RL applied to WA, the researchers Azak and Bayrak[5] implemented learning agents for decision-making in TEWA problems of C2 systems. The goal was to optimise the performance in decision-making for TEWA problems of multi-armed platforms. They used *Q*-learning to solve the WA problem and, after training, the agents learned how to coordinate with other agents and how to select a threat to engage in any state of the defence system.

They concluded that the agents benefitted from coordination and could complete scenarios more successfully than independent agents could. In addition, their default *Q*-learning algorithm implementation for WA without coordination was also successful, and agents became more mature with sufficient training data. Although coordination seemed to improve agents' performance, it is claimed that it was not a sufficient experimental achievement.

RL is particularly powerful in its role as a Markov decision process (MDP). In an MDP, the future state of a system is only a function of the state space (the set of all possible states) of the system in the present. Thus, MDPs provide a mathematical framework for modelling decision-making when outcomes are partly random and partly under the control of a decision-making agent (human or artificial). MDPs are useful for studying a wide range of optimisation problems solved via dynamic programming (DP) and RL.[6] The fact that RL can be seen as an MDP also allows the designer of the agents to work with a system in a current state, without being concerned about how it came to be in that state in the first place.

In Figure 1, a general RL problem-solving loop is depicted. The agent receives information from its environment, usually as a sensory stimulus, which gives information about the current state of the environment.[7] The agent chooses an action and upon execution receives a reward. This reward can be positive or negative, depending on the appropriateness of the action. A negative reward follows on an action that leads to a bad outcome in terms of globally set goals. Depending on the action taken, the environment might change, thereby affecting the states and actions possible in the future.

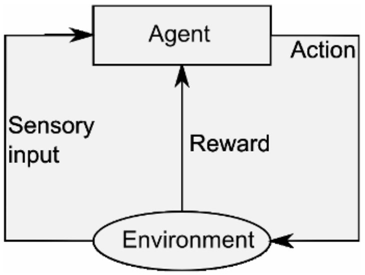


Figure 1: The RL loop

RL adapts quickly to a changing environment and indicates the measure of appropriateness of the actions taken. The agent then adjusts its memory to select a better action next time. RL algorithms therefore try to find a policy that maps states of the world to the actions the agent should take in those states. In simpler terms, the agent decides what the best action is to select based on its current state.

Figure 2 illustrates the main methods for solving the RL problem. These methods are DP, MC and TD learning. Eligibility traces can be used together with these methods to obtain a family of new methods. The methods applied to the WA problem in this article are circled.

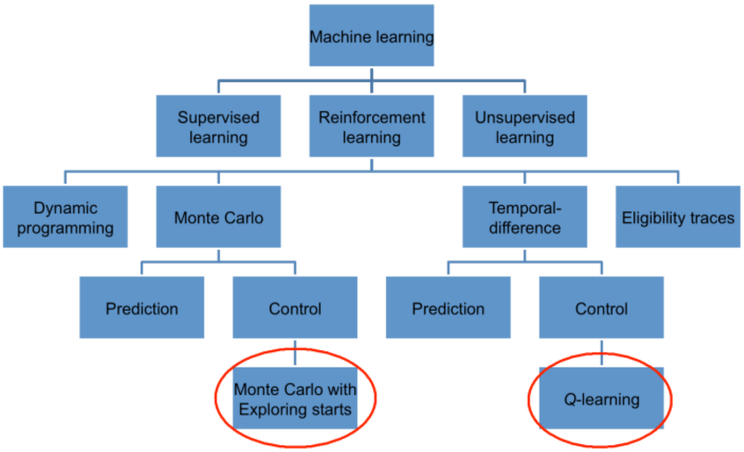


Figure 2: Outline of learning methods

The position of Reinforcement learning in the greater context

RL is not supervised learning[8]. Although RL is an online learning approach, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge), and there is no supervisor present, the algorithm learns from interaction (feedback from the environment, for example) and not from examples. This behaviour can be learned and adopted permanently, or it may be adapted continuously[3]. RL is much closer to supervised than unsupervised learning. The agent receives feedback about the appropriateness of its response. For correct responses, RL resembles supervised learning: in both cases, the learner receives information that what it does is appropriate. However, the two forms of learning differ significantly in situations in which the learner's behaviour is in some way inappropriate[9]. In these situations, supervised learning lets the agent know exactly what it should have done, whereas RL only says that the behaviour was inappropriate and (usually) how inappropriate it was.

In practice, RL is much more common than supervised learning. A teacher (especially in WA where time is of the essence) is not always available to say what should have been done when a mistake was made; and even when such a teacher is available, the learner does not always interpret the teacher's feedback correctly. In fact, often direct feedback comes from the battlefield as a consequence of the execution of a decision.

Command and control

Advances in threat technology, the increasing difficulty and diversity of scenarios, and the volume of data and information to be processed under time-critical conditions pose major challenges to tactical C2, in particular TEWA.[10] The dynamic environment in which these activities are conducted is one of high risk and high stress as it includes organised, intelligent and lethal threats. It is also uncertain due to the imprecise and incomplete nature of sensor data and intelligence, which makes unpredictable demands on operators. A military situation awareness information system must provide the commander with a clear understanding of the developing operation so that forces can be appropriately directed to meet a specified objective.[11] The C2 system is defined as the facilities, equipment, communications, procedures and personnel essential to a commander for planning, directing and controlling operations. Such a system surveys the operational environment, assesses what actions to take (i.e. aiding the TE process), and uses available resources to implement those actions (i.e. aiding the WA process).

The C2 decision cycle may be described by the most basic, well-known and widely accepted model of C2, namely Boyd's observe-orient-decide-act (OODA) loop. The OODA loop model can be represented graphically as a circular connection of the four phases of the decision cycle, as shown in Figure 3. During the observation phase, information is gathered, relevant to the decision at hand.[12]

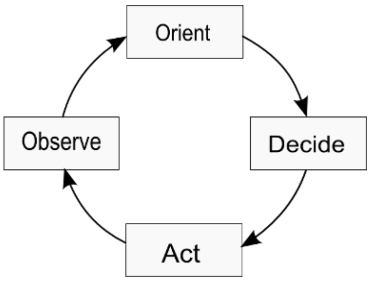


Figure 3: The OODA loop

Most of the cognitive effort during the decision process resides in the orient phase, which consists of two sub-phases: destruction and creation. A decision-making entity will attempt to destruct or decompose a problem until the sub-problems are close to situations that were experienced before, and for which the decision-maker has a plan. Problems are matched with their respective emergency plans, which are then combined or created into an overall plan of action.

If a decision-maker creates a single feasible plan (during the orientation phase), the decision phase is simply a question of whether or not to execute. The action node in Figure 3 represents the execution of a chosen course of action or plan. These actions may include a physical attack or movement, the issuance of an order, or a focus of effort on the sensor for a better observation during the next cycle of the process.

Tactical C2 can be divided into the following high-level functions:

- Target detection – mainly through sensors;
- Target tracking – use data-fusion techniques;
- Target identification – to determine true targets;
- Threat evaluation – establish the intent and capability of potential threats;
- Weapons assignment – target engagement.[13]

Threat evaluation and weapon assignment

The air defence (AD) operator assesses the tactical situation in real time and protects ground-based defended assets (DAs) against aerial threats. This is done by assigning available weapon systems to engage enemy aircraft.[14] The responsibilities associated with this environment are typically divided between several operators and decision-support systems (such as threat evaluation and weapon assignment (TEWA) that assist these operators during the decision-making processes. TEWA assigns threat values to aircraft in real time and uses these values to suggest possible assignments between weapons and observed enemy aircraft.

The TEWA process may be regarded as a dynamic human decision-making process aimed at the successful exploitation of tactical resources (i.e. sensors, weapons) during the conduct of C2 activities[10]**.** From this perspective, the role of humans is central to the TEWA functions as humans are responsible for any outcome resulting from their decisions. Hence, the type of support or aid to be provided is meant to assist, not replace, operators in their decision-making activities. As a result, the automation to be provided requires careful design. Consequently, an approach is required that models the decision-making problem and captures the cognitive demands and information requirements. The local TEWA environment is defined for AD, with the initial focus on ground-based air defence systems (GBADS),[13] and consists of two independent specialised subsystems in which the threat evaluation (TE) and weapon assignment (WA) computations are executed.

WA is a resource-management process during which weapons are allocated to engage threats in the threat-ranking list. A WA system is required to provide the fire control operator (FCO) with a proposed assignment list containing weapon system-threat associations. A WA system takes as input threat information obtained from the TE system. The assignment problem is solved given the status of DAs, weapon-systems' status and capabilities and the relevant threat information, resulting in a number of different weapon system-threat pairings. When the FCO authorises the assignment of a weapon system to a threat, the relevant weapon system must be informed. The output of the WA-model framework is a list of proposed assignments at the end of each time step.

It would be advantageous to have a system that adapts as the operator becomes more proficient at evaluating threats and assigning resources. The system should in principle then improve as the operator gains experience.

TE decision aids are suitable in addition to human cognition to make the call if an observed threat is indeed friend or foe, but with WA there is room for learning algorithms; algorithms are put in place to assist the FCO in making his/her decisions.

Modelling the weapon assignment problem

The object of the WA problem was to assign weapons effectively in order to defend a certain DA by engaging the threat as quickly or optimally as possible before it releases weapons. For our simple model, the DA was situated at the centre of the grid and was fixed. Four missile stations were used to defend the DA and, once these were placed, they also remained fixed.

The threat flew in from one of the cells and flew in a straight line towards the DA. Each weapon got a turn to shoot at the threat while it was in a specific cell (granted that the previous weapon did not succeed in its goal), according to some strategy (the policy). If the threat was eliminated, success was claimed. If all four weapons missed, the threat moved one cell closer to the DA and again the weapons took turns to shoot at the threat. If the threat reached the DA, it eliminated the DA. If the threat was on any of the weapons' positions, no shots could be fired for fear of injuring or killing the soldiers operating the missiles. In this case, the threat moved one cell closer to the DA, just as it would if all the weapons shot and missed. A typical flight pattern (assuming that all the weapons missed in each round) that we wished to avoid is illustrated in Figure 4.[15]

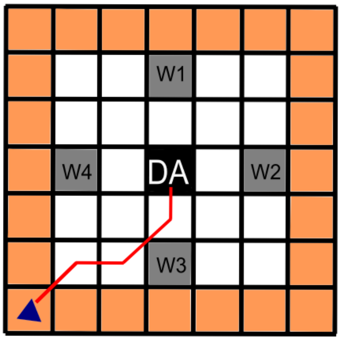


Figure 4: Flight pattern of threat eliminating DA

States and actions

Our research revealed that the finer the world-map grid, the more accurately the position of the threat could be calculated. In the terminology of RL, this aided us in defining states. Decisions were based on the position of the threat, the position being the state. In an X × X grid, there would be X^2 - 5 (X^2 minus four weapons and the DA) different states that the threat could be in at any given time step within the episode where action needed to be taken. An episode started the moment an enemy plane flew onto the grid and ended when someone (the enemy plane or the DA) was eliminated. The actions were the shooting order and in this case it was a numbered sequence, depicting which weapon shot when, for example {3,2,1,4}. With four weapons, there were 4! = 24 different actions or shooting orders.

Rewards and penalties

The WA problem was formulated as an undiscounted ($\gamma = 1$), episodic, finite MDP. The discount parameter (γ) determined the importance of future rewards. A factor of 0 would make the agent "opportunistic" by only considering current rewards, while a factor approaching 1 would make it strive for a long-term high reward. With this discount parameter, the agent tried to select actions so that the sum of the discounted rewards received over the future was maximised. It was appropriate to discount for continuing tasks and not for episodic tasks[3]. Future rewards were taken into account more strongly and the agent became more farsighted. The problem was episodic because the agent-environment broke naturally into episodes – a threat flew in and was either shot down or shot down the DA. Lastly, the problem was finite because it was an MDP with finite state and action sets (there were 24 different actions).

The system was encouraged to eliminate the threat as quickly as possible by varying the rewards given for certain cases. These different rewards are summarised in Table 1

Case	Reward
Weapon 1 eliminated threat	+3
Weapon 2 eliminated threat	0
Weapon 3 eliminated threat	0
Weapon 4 eliminated threat	0
All four weapons missed	-1
DA is reached	-5

Table 1: Obtaining the rewards

The analytical solution is that the weapon closest to the threat fired first, as we believed the first weapon to be of vital importance. For this model, we assumed that the probability of a hit was inversely proportional to the distance. The agent did not control this; therefore, it was part of the environment. We rewarded the first weapon for eliminating the threat, because we wanted to eliminate the threat as soon as possible. If the second, third or fourth weapon eliminated the threat, we did not penalise these weapons, because it was not “wrong”; it was just not what was expected. By rewarding only the first weapon, the algorithm was encouraged to use the weapon with the highest chance of eliminating the threat. If all four weapons missed, we penalised that particular state-action pair by giving it a reward of -1. When the DA was reached, a reward of -5 was given. Experiments were conducted with different reward schemes and it was found that it did not matter what the values of the rewards were; what mattered was the size of the rewards in relation to each other. Better results were obtained when the absolute value of the penalty for eliminating the DA was greater than the reward for eliminating the threat.

The policy

In our investigations, the policy was a function of the state-action pair (*s, a*). Our initial policy was to assume the weapons shot in numbered ordering, from 1 to 4. This was not a good policy, seeing that it did not take into account where the threat was located at that particular time. We wished to iterate and improve this policy until we had one that agreed with our intuition, which was that the weapon closest to the threat should fire (that weapon also had the highest probability to eliminate the threat). For this simple problem, the solution was known and this fact was used to evaluate the RL algorithm.

Solving the problem

Starting at the upper left point on the grid, an episode of WA was simulated. Starting at the next point on the grid, the following episode was simulated, and the next, until the end of the grid was reached, thus ensuring that all states were chosen as starting positions. This position was the current state of the threat. The distance from the threat to each of the four weapons was calculated, and based on that, a kill probability P_{kill} was assigned to each weapon using a lookup table. The calculation of P_{kill} was part of the formulation of the WA problem, and not of the RL algorithm.

The action volume of a weapon can be represented as a circle around the weapon. If the threat is within this circle, it will be shot down according to some probability, *e.g.*, in the circle and close to the weapon, it has a higher probability of being shot down than if it was on the edge of the circle.

Shifting the parameters

In the MC implementation, $\gamma = 1$ was used as a rule throughout all the simulations. If $\gamma = 1$ in the *Q*-learning environment, the *Q*-values diverged.[16] This was confirmed experimentally, but the experiment itself will not form part of this discussion. *Q*-learning learns the expected action-values *Q(s, a)* of taking action *a* in state *s*, then continues by always choosing actions optimally.

The learning rate parameter (α) limits how quickly learning can occur. In this specific algorithm, it directed how quickly the *Q*-values could change with each state-action change. A factor of 0 caused the agent not to learn anything, while a factor of 1 caused the agent to consider only the most recent information. If α was too small, learning was slow; and if α was too large, the algorithm might not converge.[17]

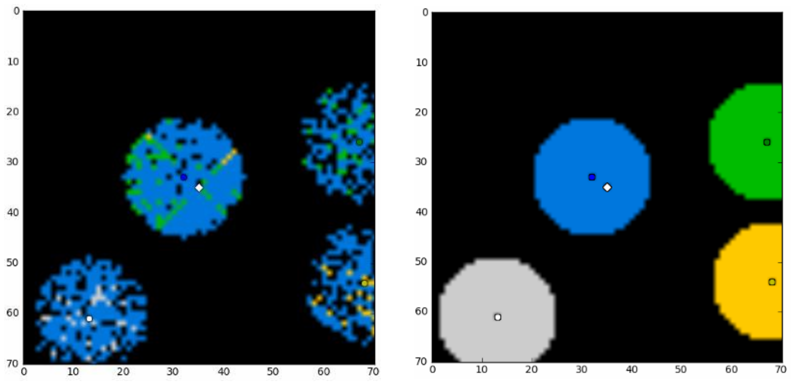
A policy that always chooses the best action is known as a "greedy" policy. Following a greedy policy most of the time, but with a small probability ε randomly selecting an action independent of the action-value estimates, is known as an ε-greedy policy. An ε-greedy policy explores all state-action pairs as the number of runs goes to infinity. For the purpose of the study, a greedy policy was used.

Monte Carlo results

The considered grids were 71 × 71 grids with randomly placed weapons. The reason for the seemingly odd choice of grid size was so that the DA could sit at precisely the centre of the grid. The firing distances as well as the number of simulations per state were changed.

The four weapons were each marked with a coloured dot on the grid and the DA was shown as a white diamond in the centre. The blue area corresponded to that part of the grid covered by weapon 1 (blue dot). The green area corresponded to the part of the grid covered by weapon 2 (green dot); the yellow area to the part covered by weapon 3 (yellow dot); and the grey area to the part covered by weapon 4 (white dot). The black spots on the grid were the states where no shots were fired. Recall that no shots were fired if the threat was in the same position as one of the weapons, or if the threat was outside of the weapons' firing range

Figure 5 shows two images depicting the simulation results for a 71 × 71 grid with maximum firing distance equal to 12. Figure 5(a) shows the situation at the start of the simulation and Figure 5(b) shows the simulation after completion.



(a) Before simulated learning

(b) After simulated learning

Figure 5: A 71× 71 grid with 50 simulations per state and firing distance 12

Figure 6 shows the simulation results for a 71 × 71 grid with a maximum firing distance of 20. The spots of (different) colour seen inside the circles are there because that particular state was not visited by a successful weapon and was thus a remnant from a time when the original policy designated that state as being shot at by a different ("wrong") weapon. This was mainly due to the larger firing distance (with respect to a firing distance of 12) as well as the fact that weapon 2 overlapped with both weapon 1 and weapon 3's firing circle.

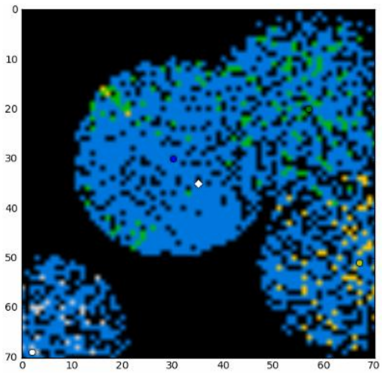
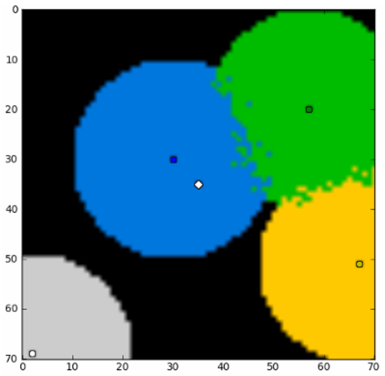
	
(a) Before simulated learning	(b) After simulated learning

Figure 6: A 71 × 71 grid with firing distance 20

Note that these erroneous actions taken were not "wrong"; they were just not the best actions that could have been taken in that situation. A possible reason for these discrepancies is the fact that, even though the "wrong" weapon shot and eliminated the threat, its *Q*-value was probably still larger than other state-action pairs with negative *Q*-values.

Temporal-difference results

As with the MC examples, the firing distances were changed as well as the number of simulations per state.

We chose to keep $\varepsilon = 0$ and vary the γ- and α-values. Through experimentation, the best values for γ and α were found to be in the ranges $0.2 \leq \gamma \leq 0.7$ and $0.1 \leq \alpha \leq 0.2$.

		Error % for various firing distances	
γ	α	12	20
0.2	0.1	4.939%	11.287%
0.3	0.1	4.999%	11.645%
0.4	0.1	**4.483%**	11.168%
0.5	0.1	4.701%	**9.919%**
0.6	0.1	6.784%	10.137%
0.7	0.1	5.297%	10.950%
0.2	0.2	9.601%	18.231%
0.3	0.2	9.502%	16.425%
0.4	0.2	7.499%	15.314%
0.5	0.2	8.411%	16.128%
0.6	0.2	7.935%	14.779%
0.7	0.2	8.272%	14.204%

Table 2: Error percentages

The minimum errors for the three firing distances occurred when $0.4 \leq \gamma \leq 0.5$ and $\alpha = 0.1$. Using these parameter combinations, we obtained the following results. The same weapon layouts and firing distances were used as for the 71 × 71 grids in the MC section.

From Figure 7 it can be seen that learning for a 71 × 71 grid with firing distance 12 was very slow. This could be due to the small α -value (learning rate). Figure 8(a)–(d) also shows a slow learning rate for a firing distance of 20.

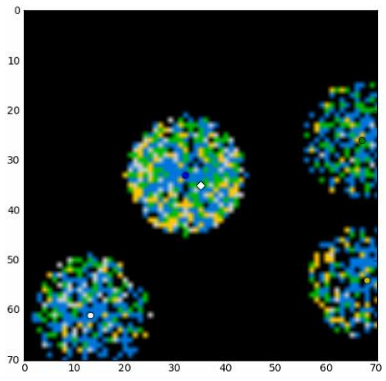
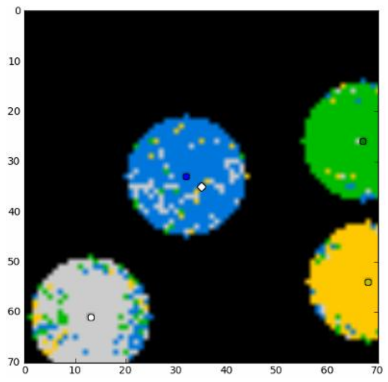
	
(a) Before simulated learning	(b) After simulated learning

Figure 7: A 71 × 71 grid with firing distance 12

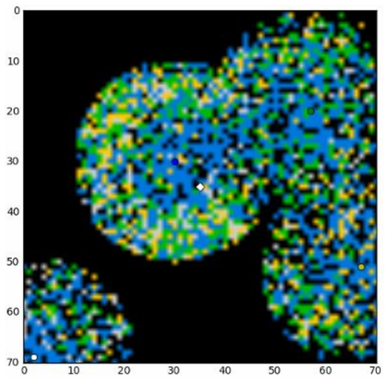
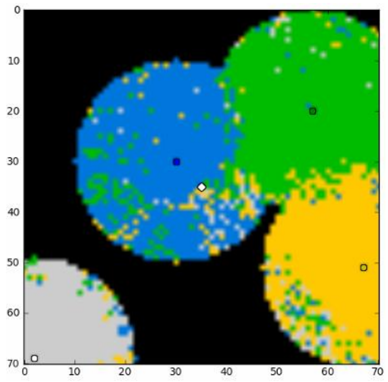
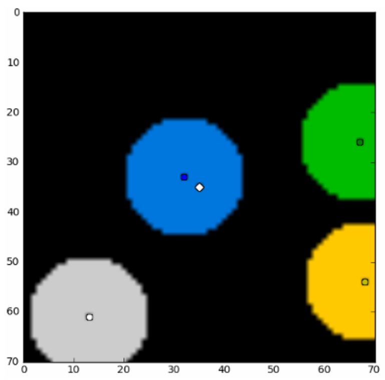
	
(a) Before simulated learning	(b) After simulated learning

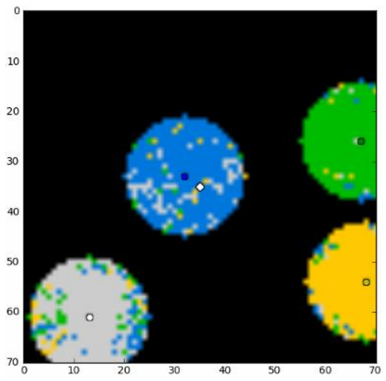
Figure 8: A 71 × 71 grid with firing distance 20

Discussion of results

Figure 9 and Figure 10 show the side-by-side comparison for each of the completed simulations of the 71 × 71 grid examples.

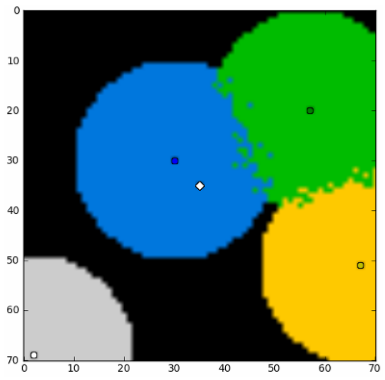


(a) Monte Carlo ES

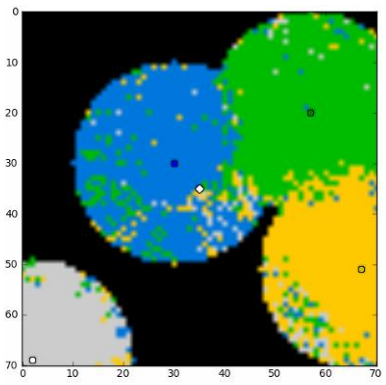


(b) TD *Q*-learning

Figure 9: A 71 × 71 grid with firing distance 12



(a) Monte Carlo ES



(b) TD *Q*-learning

Figure 10: A 71 × 71 grid with firing distance 20

The results are visibly different with TD (*Q)* faring worse than MCES. According to Sutton[3], the learned action-value function, *Q*, can be shown to converge to the optimal action-value function, *Q**, with a probability of one. This is clearly not what was happening here. The reason for this is that the best parameter-value combination had not been found yet. Table 3 shows a comparison between the error percentages for both algorithms.

	Error % for various firing distances	
	12	20
MCES	0.258%	1.508%
TD (*Q*)	4.483%	9.919%

Table 3: Error percentages

Conclusion

The aim of the study on which this article is based, was to evaluate and compare two RL algorithms to see which would yield promising results on the simplified version of the WA problem. An MCES control algorithm and the off-policy TD-learning control algorithm, *Q*-learning, was applied to 71 × 71 grids.

Simulations were run with firing distances of 12 and 20 grid units. It was found that the smaller the firing distance relative to the grid size, the longer the episodes were and the longer the simulations took. However, a smaller firing distance incurred a smaller error percentage. The bigger the firing distance, the greater the chance of overlapping between the weapons and the bigger the error percentage.

For the MCES algorithm a fixed discount parameter of $\gamma = 1$ was used. This algorithm performed exceptionally well and incurred only minor error percentages. When applied to a 71 × 71 grid with 200 simulations per state, the MCES algorithm took nearly twice as long as the *Q*-learning algorithm with a firing distance of 12, and nearly three times as long with a firing distance of 20.

Setting the discount parameter to $\gamma = 1$ caused the *Q*-values to diverge in the *Q*-learning algorithm. A greedy policy was followed while the learning rate (α) and the discount parameter (γ) were varied. It was determined experimentally that $0.1 \leq \alpha \leq 0.2$ yielded better results than larger α-values. These α-values were used with different γ-values and it was found that medium to large γ-values worked best. It was found that with a greedy policy, $\alpha = 0.1$ gave adequate results when applied along with $0.4 \leq \gamma \leq 0.5$.

Even though the MCES algorithm was considerably slower, it outperformed the *Q*-learning algorithm in all examples considered. The problem of simulation time could be rectified by code and hardware optimisation. We thus conclude that

RL, especially the MCES algorithm, is a promising field to consider in the solving of the WA problem. Combining the techniques investigated and tested in this work with other techniques in AI and modern computational techniques may hold the key to solving some of the problems we now face in warfare.

Future work

The problem of considering both TE and WA within the context of OODA is non-trivial and future work must focus on bringing these elements together and considering the computational burden that would result. The obvious question that comes to mind then is how one could find optimal policies that are developed in timely fashion to support the commander during an operation.

Another suggestion for future work would include designing a more complicated grid by including different terrains and "obstacles" such as mountains and valleys. Time delays for weapons could also be added, or even the constraint that certain weapons have limited ammunition. Another constraint could be that certain weapons can shoot further than others can, where we currently have a constant firing distance across the board in a given problem. Another suggestion would be to consider the case where multiple weapons shoot at the same time.

A possible improvement on the *Q*-learning algorithm would be to follow an ε-greedy policy and vary ε along with γ and α, as opposed to only using a greedy policy.

Endnotes

[1] Smith, R. *The utility of force: The art of war in the modern world,* Knopf, New York, 2007.

[2] Russell, SJ & Norvig, P. *Artificial intelligence: A modern approach,* 2nd edition. : Pearson Education, 2003.

[3] Sutton, A & Barto, R. *Reinforcement learning: An introduction*, 1st edition. The MIT Press, Cambridge, Massachusetts, 1998.

[4] Naidoo, S. *Applying military techniques used in threat evaluation and weapon assignment to resource allocation for emergency response: A literature survey*, 2008. <http://www.orssa.org.za/wiki/uploads/Johannesburg/Naidoo2008.pdf> Accessed on 17 June 2009.

[5] Azak, M & Bayrak, A. "A new approach for threat evaluation and weapon assignment problem, hybrid learning with multi-agent coordination". In: *23rd International Symposium on Computer and Information Sciences (ISCIS)*, Istanbul, 2008, p.1-6.

[6] Puterman, ML. *Markov decision processes: Discrete stochastic dynamic programming.* Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, Wiley, New York, 1994.

[7] Engelbrecht, AP. *Computational intelligence: An introduction,* 2nd edition, Wiley, Chichester, 2007.

[8] Hu, J, Sasakawa, T, Hirasawa, K & Zheng, H. “A hierarchical learning system incorporating with supervised, unsupervised and reinforcement learning”. In: Liu, D. (ed), *Advances in neural networks: ISNN 2007.* Springer-Verlag, Berlin, Heidelberg, 2007, p.403-412.

[9] Gasser, M. *Introduction to reinforcement learning,* 2009. <http://www.cs.indiana.edu/~gasser/Salsa/rl.html> Accessed on 8 April 2009.

[10] Paradis, S, Benaskeur, A, Oxenham, M & Cutler, P. “Threat evaluation and weapons allocation in network-centric warfare”. In: *2005 7th International Conference on Information Fusion (FUSION),* Piscataway, NJ, USA, IEEE, 2005, p.1-6.

[11] Rosenberger, JM, Hwang, HS, Pallerla, RP, Yucel, A, Wilson, RL & Brungardt, EG. “The generalized weapon target assignment problem”. In: *10th International Command and Control Research and Technology Symposium,* McLean, VA, 2005, p.1-12.

[12] Brehmer, B. “The dynamic OODA loop: Amalgamating Boyd’s OODA loop and the cybernetic approach to command and control”. In: *10th International Command and Control Research and Technology Symposium,* McLean, VA, 2005, p.1-15.

[13] Leenen, L & Hakl, H. *TEWA: Integrating Threat Evaluation with Weapon Assignment*. Council for Scientific and Industrial Research (CSIR) DPSS-SM-EDERIMSDS-032 Rev 1, January 2009.

[14] Roux, JN & Van Vuuren, JH. Real-time threat evaluation in a ground based air defence environment. *ORiON* 24/1. 2008. 75–101.

[15] Mouton, HS. “Reinforcement learning: Theory, methods and application to decision support systems”. Master’s thesis, University of Stellenbosch, 2010.

[16] Watkins, CJCH & Dayan, P. “Technical note *Q*-learning”. *Machine Learning* 8. 1992. 279–292.

[17] Cline, BE. *Tuning Q-learning parameters with a genetic algorithm*. 2004. <http://www.benjysbrain.com/ThePond/Tuning.pdf> Accessed on 8 April 2009.