Serpentine
Artificial Intelligence Association

Technical report

MIT Battlecode 2021 – Viper

Bas Paardekooper
Pieter Voors
Viktor van Bilsen
Wolf van der Hert
viper2021@serpentineai.nl

January 2021

Abstract

Battlecode is a yearly competition hosted by MIT, every year with a different 'game'. This report will analyse the strategy and code of team Viper, one of the participating teams from Eindhoven University of Technology.
This year the game is about movable units, each with a special power, who want to win 'the election' of their political party. A team can win if they, after 1500 rounds, have bought the most votes, with their currency 'influence'. But it can also be won if all movable units and the enlightenment centers of the enemy team have been defeated.
Four main strategies were implemented by team Viper. Information between units can be shared using flags across the map to the enlightenment center. So, a flag system was implemented to share information with other units across the map, in order for new units/robots to get specific tasks and their location. Some units had the main task to stay alive and run away from robots of the opposite team. Another strategy was to send a specific kind of robot, the politician, to neutral or enemy enlightenment centers to concur and convert them. One kind of robot is 'cheap' to make, the muckrakers, so they were used to eliminate enemy robots and to explore the map.
With this strategy team Viper came in 82nd place out of the 559 teams, with an Elo score of 1336. There were a total of 150 wins and 144 losses for Team Viper.

Contents

1 Introduction

From January 4 to January 30, 2021, four members of Student Team Serpentine formed team 'Viper' and competed in the MIT Battlecode 2021 competition [1], hosted by MIT [2]. Their competitions are based on 2D grid - games where bots compete to find out what team can create the best AI agent in one month. Another Serpentine Team 'M.A.R.S.' participated to this competition. Serpentine participated previously in Battlecode 2020, with teams *Noodles* and *SnakeEyes*. [3][4]
The game this year consisted of 1500 rounds, in which each unit on the map executes code individually. On a randomly selected map - a 2D grid with areas with different terrains (Determining the speed in which units can move and take actions) - a map-dependent-number of Enlightenment centers (EC) are situated, of which some are owned by team blue, some by team red, and some are neutral. Each turn a certain cool-down was met, these stationary buildings can spawn one unit in a free tile next to the building, which costs 'influence'. Influence represents the currency in the game. A certain 'HP' value is given to spawned units which is called 'conviction', and is determined by the amount of influence used to produce the unit. An EC can also raise a flag that is visible to all units that can see it or that know the EC ID.
The three types of units in the game are called *Politicians, Slanderers* and *Muckrakers*. All units can walk around the map and raise a flag with a specific color. This, together with the EC flags being visible to all units, is the only way of communication, as units cannot 'talk' to each other, and have a limited range in which they can sense other units and have another limited but different range in which they can indicate the type of unit within that range. A politician can use conviction to 'empower' enemy units or ECs, which destroys the politician in the process. Empowering an enemy unit will subtract the used conviction from their current conviction. If this value turns negative, the enemy unit will become friendly if it is a politician or EC, and will be destroyed if it is a slanderer or muckraker. If a friendly unit is hit by this action, their conviction is increased up till their original influence cap. Slanderers are units that are identified as politicians by the enemy (except for enemy muckrakers), and 'spread lies' in the world, to gain influence for their team. Muckrakers can 'destroy these lies', which is an action they can take when an enemy slanderer is within their identification range that will destroy the slanderer. The sensing range of the units is larger than the identification range, and the muckraker has larger ranges than the other units. The appearances of the units can be seen in figure 1.
Each round, the teams can spend influence to attract the round's 'vote'. The team that used more influence to attract a vote wins the vote for that round. The team with the most votes after the 1500 rounds wins. Another way to win is to take over all enemy ECs and eliminate all enemy units.

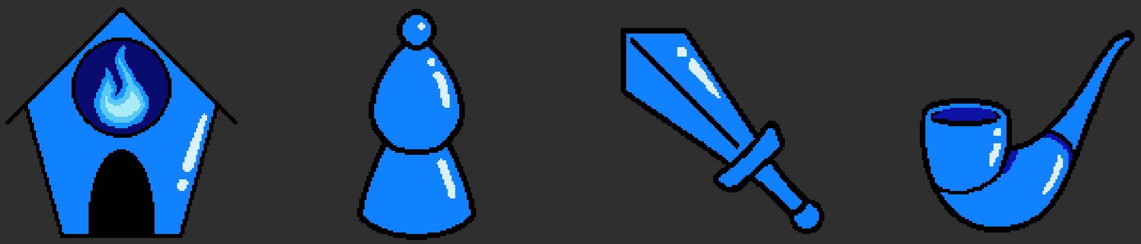
(a) Enlightenment center (b) Politician (c) Slanderer (d) Muckraker

Figure 1: Textures of the units used in the game.

2 Strategy

The strategy used is an offensive strategy. The general idea of the strategy is to first create slanderers to get more income, which try to avoid enemies to stay alive, have politicians take over neutral ECs and have muckrakers search for and block off the enemy ECs by standing around them, while attacking enemy slanderers. To achieve this, each type of agent has its own purpose and thus its own strategy.

2.1 Flag system

Flags can be used in the game to share information with other agents across the map. This system is used to inform each other about the locations of enemy ECs, as well as give new assignments to units. Whenever an agent spots a neutral or enemy EC which was not yet known, it raises its flag to notify the friendly EC(s) that an EC was found. This friendly EC then copies the flag to inform other friendly agents of the location of the newly found EC.

2.2 Slanderers

The slanderers are responsible for generating more resources. To achieve this, they need to make sure to stay alive. Most importantly, they should avoid enemy muckrakers, as they can be easily destroyed by them. Therefore, when slanders spawn, they choose a random ordinal direction and keep walking in this direction in order to walk away from the base. This allows for an easy way to create a spread of the slanderers in multiple directions, while also being easy to create a program for compared to non-ordinal directions. In order to avoid muckrakers, whenever a slanderer notices an enemy muckraker in its vision, it always tries to move directly away from it. It then continues to do so (ignoring its ordinal direction), as the direction the muckraker came from is likely to be the direction an enemy ECs is in and thus also the direction other muckrakers may come from, which should be avoided.

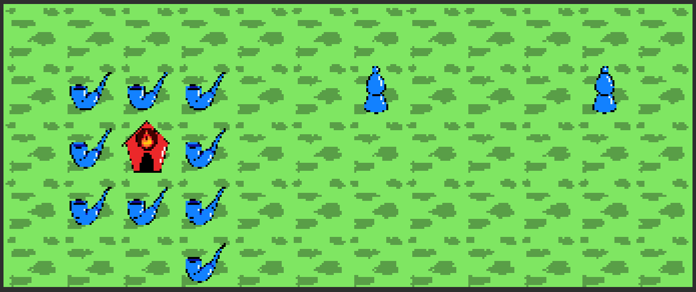

Figure 2: Muckrakers circling the enemy EC, with politicians coming in to capture the EC.

2.3 Politicians

The main purpose of the politicians is to capture neutral ECs, as well as empower enemy units. Therefore, when a politician gets the location of a newly discovered EC, the politicians move towards this EC, to then take over the EC when they are close to it, as capturing new ECs provides more resources and gives a new place to spawn units from. An example of the politicians moving towards an enemy EC can be seen in Figure 2. Whenever a politician spots an enemy unit, it checks whether it is beneficial to capture this enemy. If so, it speeches with a range as small as needed to affect this enemy, so that the speech is as effective as possible, losing less resources on the base cost of speeching to a unit. Apart from this, it may sometimes also be beneficial to speech to a friendly EC. When muckrakers have destroyed enemy slanderers, they provide a boost to the power of speeches of the politicians. This means that the ECs can create politicians with some amount of resources assigned to them, whereas the politician can then speech to the ECs, putting more than its original resources back into the ECs.

2.4 Muckrakers

Muckrakers are cheap and effective against slanderers. Therefore, whenever a muckraker spots an enemy slanderer, it destroys it. If the slanderer runs away before the muckraker can attack it, it will run after it in an attempt to destroy it. Another responsibility of the muckraker is to explore the map and find (enemy) ECs. The muckrakers are a good pick for this, as they are cheaper to create than politicians and have a larger vision range and a higher movement speed than slanderers. Lastly, when muckrakers find an enemy EC, apart from raising a flag to notify others of the location, they also attempt to block the enemy EC by standing around it. An example of this can be seen in Figure 2. When there are agents all around an EC, the EC cannot spawn any agents anymore, effectively making the EC useless other than for generating resources. Whenever a muckraker arrives at an enemy EC that is already fully blocked however, it just continues exploring, as there is then no use in it attempting to block as well.

3 Implementation

This section will detail how the strategies were implemented and what algorithms were used. Some functions and implementations are shared between multiple or all units, these will be explained separately. All other unit specific implementations will be discussed per unit.

3.1 Flag system

The only way to communicate between units in this iteration of Battlecode is by using flags. Every unit can raise a flag, in the form of a 24-bit integer, which can be observed by other units. A unit can request the current state of another unit's flag by using their ID while they are in their sensor range. Apart from this, every unit can always get the flag of ECs they have the ID of and the other way around. This feature was used to make the EC the center for communication between our units.
The flag system is used for signalling locations on the map. The locations, however, range from (10 000, 10 000) to (30 000, 30 000), which is too big of a number to encode in the 24 bits that can be used for one flag's color. The map itself, however, it at most 64 by 64 in size. Thus, you would want tho send the location relative to the lower left corner of the map. However, you do not know where the corner of the map is. You can, however, store the value assuming the map is of size 128 by 128, with the lower left corner at a multiple of (128, 128). This way, it is certain that the map lies within this square and by taking any position modulo 128, we get a position relative to this custom chosen lower left corner, making that we can use the locations within the playing field. [5] This way of encoding locations used only 14 bits in total for the x and y and so left 10 bits of room for extra information. In this extra space the meaning of the location was also sent. The first 500 out of the 1024 possible integers were used to indicate a neutral EC along with its conviction so a politician can be made that is just strong enough to take over the neutral EC with minimal influence loss. Besides that, other integers in the rest of the range were used to indicate that an enemy EC is at that location or that the EC in that location has been taken over.

3.2 AI-level

A system that was dubbed *AI-levels* was used to make units execute differing code based on some variable, so each unit had this variable to represent the state the unit is in. This was essentially just an integer that showed in what state the unit was in. Each integer encoded for a different type of behaviour, such as defensive and offensive for politicians. This could change depending on a units surroundings or flag from an EC.

3.3 Enlightenment center (EC)

The enlightenment center is an immobile unit that can vote and produce other units. It does this by spending influence. ECs get influence passively or by building a slanderer that can use their embezzle ability to generate influence for them.

3.3.1 Voting

The EC used a fairly rudimentary voting strategy. This was mainly used as it was the first thing that could be reasonably implemented and it was never replaced afterwards. In this strategy the EC did not look at what the opponent was doing with voting, it simply voted based on the following parameters. Firstly, an EC would only vote if their side had less than 751 votes. This was done because it does not matter how many votes you have beyond the halfway point of the 1500 total votes. Secondly, an EC would only vote if it had more than 350 influence or if it was later than round 500. These values were arbitrarily chosen based on the fact that it should not start voting too early when influence is still very valuable for possible politicians or slanderers. When it was decided that an EC would vote, it would simply vote 5% of their current influence. This was also arbitrarily chosen based on the fact that not too much influence should be spent, while the bid still has to have a chance to win (To be a higher value than the enemy vote). This kind of strategy does lead to the fact that every EC votes, which did not become a problem as when multiple EC vote, only the highest bid will go through and the other bids will be discarded and the influence of those lower bids will be completely refunded.

3.3.2 Unit production

For unit production a queue system was used based on a first in first out system. Every time an EC could build something it would build the first thing in the queue. If the EC did not have enough influence to build the first item in the queue it would be moved to the back of the queue and instead the next item in the queue would be looked at, this was done at most 5 times. If this swapping indeed happened 5 times, a muckraker of strength 1 would be built as this is nearly free and could still be very useful.
Not only the type of unit but also other variables were stored in this queue. For each item besides the type of unit also the amount of influence to build it at and the preferred direction relative to the EC to build the new unit at could be given. When the amount of influence was not specified it would be determined at the time of building and if the preferred direction was not specified it would be built in any open possible spot.
Multiple ways to fill this queue were used. To start, the simplest way was to just add a unit of a certain type to the queue every certain number of rounds, this was used for one of the ways to add slanderers and muckrakers to the queue. The second way is similar to the first, adding a unit to the queue every certain number of rounds, but the amount of rounds between each time is decreased so more are added later on, this was used for politicians. Politicians could also be added to the front of the queue when exponential influence gain was possible, this happened when the politician empower multiplier was high enough. The EC also looked at their environment to determine what to add to the queue. If there had not been any enemies close for a while then more slanderers would be added to the queue. Conversely, the more enemies were nearby the more priority was given to adding politicians to the queue. Lastly, when the queue is empty it adds a 1 strength muckraker to the queue for the same reason as stated earlier. This 1 strength muckraker instead becomes adding a politician to the queue later on in the match.

3.3.3 Communication

The enlightenment centers were used as the hubs for communication between the units as all units could see the flags of ECs they have been close to regardless of distance and the other way around. This allowed for two way communication between units no matter how far they are apart.
The EC would look through as many of the flags of the units they knew per round as they could up to a computation limit, the others would be done in other rounds. This way the ECs could look for all friendly flags even with the limited computation time. When a flag was found that would indicate a neutral or enemy EC location for example this flag would be stored. Then every round the EC would cycle through the flags they had at that moment. This list consisted of only one enemy EC and all known neutral ECs. When a flag was found indicating that an enemy or neutral EC had been taken over this location would be cleared from the list so units would stop going there.

3.4 Muckraker

Muckraker units have two types of behaviour. The first being looking for enemy or neutral ECs and and enemy slanders and the second being blocking enemy ECs. The muckrakers would start in scouting mode and would switch to blocking mode when they had spotted an enemy EC or when they had received a signal from their EC concerning the location of an enemy EC. They would switch back when said enemy base had been taken over or when all spots were already taken up by blocking muckrakers.

3.4.1 Scouting

When a muckraker is in scouting mode it will go to random places in hopes of finding an enemy or neutral EC or enemy slanderer. If an enemy slanderer is found it will try to destroy it by following them. When an EC is found they set their flag to that location with the appropriate extra information to indicate either an enemy EC or a neutral one. In the latter case also the amount of influence this neutral EC has will be encoded in the flag. In the case that an enemy EC is found, muckrakers switch to the blocking AI-level.

3.4.2 Blocking

Muckrakers that are trying to block will go the location of an enemy EC and block their possible places to create a new unit, essentially shutting down the enemy EC. Besides this stopping of creating new units it also allows them to destroy any possible slanderers that are nearby. Muckrakers will switch back to the default scouting behaviour when the EC has been taken over or when all places are already blocked when one arrives as not to create too much of a pile up of units.

3.5 Slanderer

Because of the nature of slanderers, slanderers can only run away from threats. When they are created a random direction is picked to run towards so they do not stay near the EC. If an enemy is spotted they try to run away by running in the opposite direction of the enemy. In the case that a slanderer lives long enough to turn into a politician, it will move randomly until a location of an enemy base is received from their EC. As soon as this happens they will go there and try to take over this base. It will also always empower when an enemy is found at a radius that is just enough to hit this enemy unit as not to waste too much conviction over friendly possible units. This was supposed to be a first implementation but was never changed in the end.

3.6 Politician

Viper's politicians have three possible states. The first is offensive, this consists of moving to random locations trying to find enemies and empowering when one is found and going to enemy enlightenment centers to try to take them over. The second state is conquering, this is used for taking over neutral enlightenment centers. The last state is defensive behaviour, walking around friendly ECs, empowering when an enemy is found.

3.6.1 Offensive

Politicians by default have the AI-level corresponding to offensive behaviour. They try to look for enemy bases or go to one when they receive the location of one. When an enemy is found they always empower if it is possible, they do this with a radius exactly enough to hit this enemy as not to waste too much conviction.

3.6.2 Conquering

Politicians that are built with the specific intent of taking over a neutral EC are given this behaviour by the use of a flag. They will go to the indicated position and take over the neutral base. This will almost always be possible as politicians will only go to this AI-level if they have enough influence to take over the neutral EC in question. If one arrives at the location and it has already been taken over it sets its flag so that this location is cleared in the EC and the politician switches back to offensive AI.

3.6.3 Defensive

When a politician is first built it will look around for friendly politicians showing the flag indicating that they are defending that EC. In the case that a politician cannot find enough it will become a defensive politician itself, unless it can conquer a neutral EC in which case it will go do that. Defensive politicians walk around to random locations around the enlightenment center looking for enemies to use empower on.

3.7 Pathfinding

All of the units apart from the ECs are able to move around. Whenever a unit needs to walk in an ordinal direction, this can simply be implemented by making the unit walk in that direction any time the unit is ready to move and the next cell in that direction is not occupied by another unit.
Some units, like the politician, sometimes have to move to a specific location. To do this, they could simply continuously fetch the ordinal direction that is closest to the direction from their current position to the destination, which can be achieved from the Battlecode framework, to then walk in that direction. While this would work, this could result in walking through dense cells (which make the unit walk slower) or in standing still against another unit waiting for them to move out of the cell they want to move into. Therefore, a short-range pathfinding algorithm was used, an example of which is partly illustrated in Figure 3. For all cells on the border of a small range around the unit (illustrated by the orange border), the Hamiltonian distance from that cell to the destination is computed (illustrated by the green arrow for two border cells), representing an estimate of the time it would take to walk to the destination from the circle edge cell. Then, for each cell, the minimum time it takes to walk from the unit position to that cell, taking into account the density and unit-occupation of the cells is added to this using Dijkstra's algorithm[6] (illustrated by the orange arrows for two border cells, where blue tiles take two turns to pass instead of one). This then results in each circle edge cell having an estimate of the time it would take to move to the destination via that cell (9 and 7 in the example). Then, the cell with the lowest estimate is chosen and using the results of the Dijkstra algorithm, the next cell on the optimal path from the unit towards this circle edge cell is selected as the cell to move the unit to (in the example moving right). This way, units are able to walk around other units as well as avoid small dense cell patches.
For the range of the pathfinding circle, a radius of 2 was used. When a higher radius than this was used, it was experimentally found that the computational limit was reached, which makes the unit temporarily freeze. As an as high as possible radius improves the ability to avoid dense cell patches or clusters of units better, but freezing hinders the movement of the unit significantly, we chose this radius, which is the highest radius achievable without making the bot freeze.

4 Results

4.1 Implementation Results

In Figure 4 a state freeze of a game is given, where two Viper bots competed against each other. First of all, a lot of muckrakers are gathered in the sides and corners of the map. These are muckrakers that have succeeded to explore but have not found anything, and are making it unsafe for enemy slanderers. In the bottom-right, sufficient blue slanderers have gathered and are well protected from

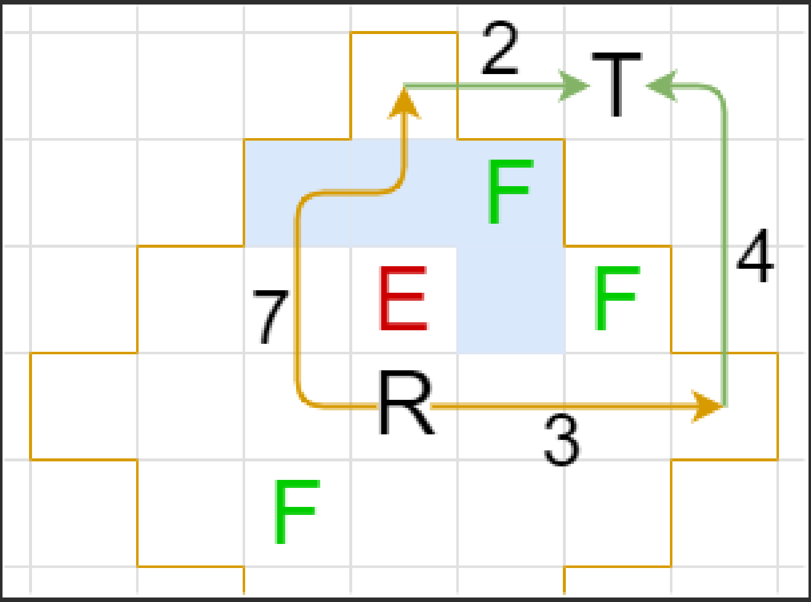

Figure 3: Illustration of an example of the pathfinding algorithm

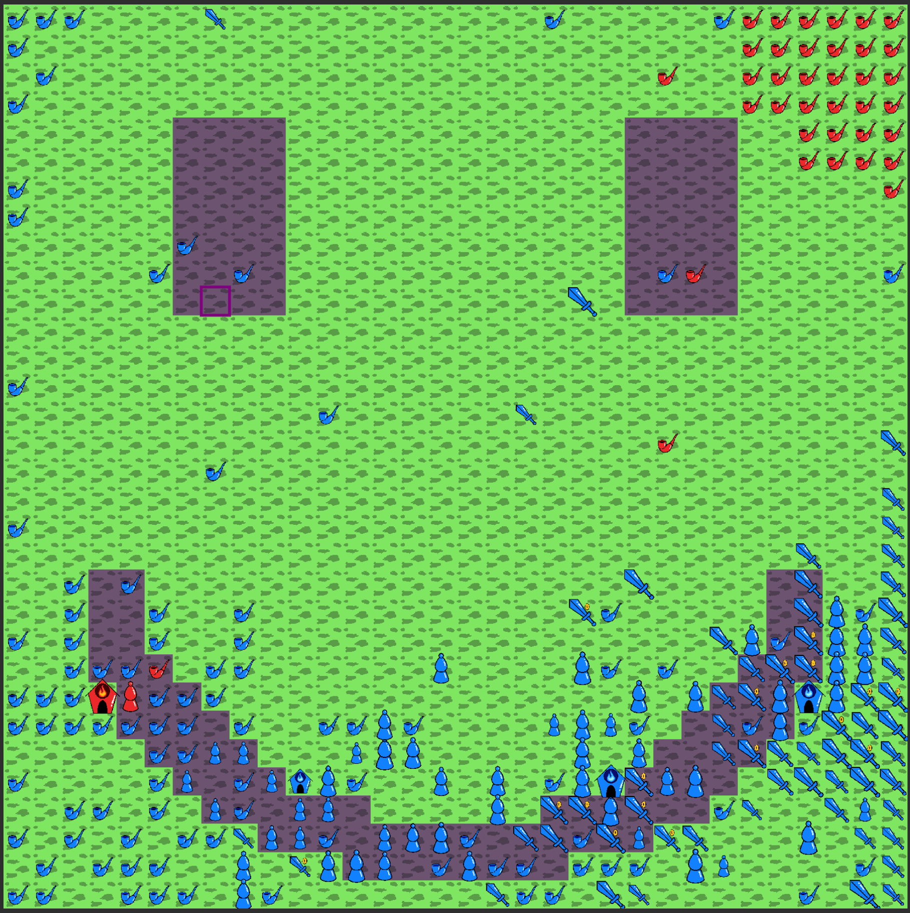

Figure 4: State freeze of a game of Battlecode.

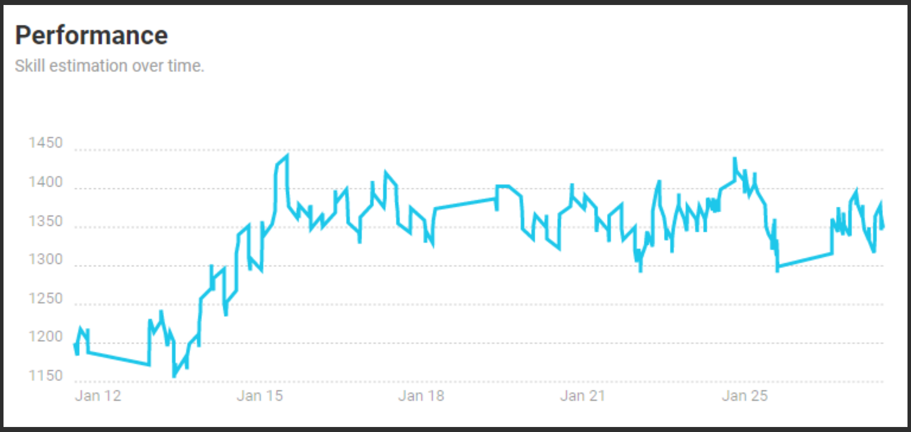

Figure 5: Graph of the skill estimation over time, representing the Elo score and thus performance of bot Viper.

the enemy muckrakers. In the bottom-left, one can see the effect of the strategy to block enemy ECs. The red EC is surrounded by blue muckrakers, making it unable to spawn new units. The blue team succeeded to take over two ECs, by sending politicians.

4.2 Intermediate Results

There were three intermediate optional tournaments, of which Viper participated in the first and last, held on January 11 and January 26, respectively. The intermediate rank Viper obtained for the first tournament was 46. The last intermediate tournament (The preliminaries) was held to determine whether international teams (Teams outside of the US) could participate in the finals, held on January 30. Viper did not make it to the finals and got a rank of 56 in the preliminaries. In figure 5, the skill estimation over the competition period can be seen. This is based on intermediate and final scrimmages and tournaments.

4.3 Final Results

Although the last submission for international teams which did not make it to the finals was one day before the preliminaries, the US teams together with the international teams that passed the international preliminaries were able to develop their bot further for the finals. International teams were allowed to improve their code (Which team Viper chose not to do), however, they would not be taken to the finals. During the finals, Viper competed against those newer bots and obtained the final results as listed in table 1.

Rank	82
Elo Score	1336
Wins	150
Losses	144

Table 1: Final results of bot Viper, as listed on the Battlecode 2021 website

5 Discussion

Viper had a quite clear understanding of how the game worked. They watched games back to look for possible improvements, which helped them in the contest.
Some strategic problems include the slanderers 'running away' from potential threats, which caused them to run into the sides of the map, and stay stuck there. If more improvements were to be made

on the code, it is recommended to make the slanderers not run into the wall or each other, so that they can move if necessary.
Another thing was that the ECs of team Viper were not well enough protected, and thus could easily be overthrown. Many games were lost due to lack of conquered ECs, and therefore it is recommended to 'guard' the ECs better with muckrakers or slanderers.
An other important part was the time challenge. More feedback could have been implemented and some strategic problems potentially could have been solved with that, if time was invested in that.

6 Conclusion

Bot Viper has a strong foundation to tackle the main obstacles introduced by the game mechanics: The communication possibilities are discovered, the weaknesses and strengths of the different types of units are found and used in a strategy set, and multiple strategies are offensive and can lead to a victory. However, there is room for more complex strategies to be built on this foundation. Team Viper has made less progress concerning other challenges Battlecode brings: The foundation was completed in a late stage, which put the team in an awkward position in time. More complex iterations on the current state of the bot would likely have given Viper a boost in the competition.

References

[1] MIT Battlecode 2021. `https://battlecode.org/` Accessed on: January 1 2021.

[2] MIT - Massechusetts Institute of Technology `https://www.mit.edu/` Accessed on: January 1 2021

[3] Grooten B.J., Genuchten D., Ronhaar M *Battlecode 2020 - Serpentine Noodles* Serpentine, 2020 `https://serpentine.ai/wp-content/uploads/2020/03/whitepaper-noodles.pdf`

[4] Voors P., Lepelaars C., Tomilin T. *Battlecode 2020 - Serpentine SnakeEyes* Serpentine, 2020 `https://serpentine.ai/wp-content/uploads/2020/03/whitepaper-snakeeyes.pdf`

[5] MIT *Battlecode lecture 4, 2021*. Battlecode lecture 4 Youtube, MIT Battlecode, 2021.

[6] Dijkstra, Edsger W. *Numerische matematik*. [*A note on two problems in connexion with graphs*]. Springer, 1:269-271, 1959.

7 Acknowledgements

This project was done within Serpentine, the student team of the Eindhoven University of Technology (TU/e) that competes in AI programming contests. We would like to thank the TU/e, as well as our industry partners VBTI and Flowserve for supporting our work.