Battlecode 2021 Postmortem

Ivan Geffner (XSquare)
Malott Fat Cats

1 Introduction

I'm a Math graduate student at Cornell University, and this is my 7th and (unfortunately) my last participation in Battlecode: **Team Potato** (2015, 13th-16th), **Felix and The Buggers** (2016, 9th-12th), **Oak's Disciples** (2017, 3rd; 2018, 4th), **Oak's Last Disciple** (2019, 33rd-48th) and **Java Best Waifu** (2020, 1st). This year I teamed up with fellow Cornell Math grads Jose Bastidas, Rodrigo Horruitiner and Ryan McDermott under the team **Malott Fat Cats** and placed fourth in the final tournament. We chose this name because Malott is the name of the Math dpt. building and Malott FC (which unofficially stands for Malott Fat Cats) is our soccer team.

Our final submission can be found here.

2 Game Overview

This year's game consists of winning an election in the midst of political turmoil.

Each team may use the following units:

- **Enlightment Centers:** It produces some *influence* every turn (the main resource of the game), and can build *Politicians*, *Slanderers* and *Muckrakers*. Whenever the enlightment center builds a unit, it can pay an arbitrary amount x of influence, and that unit is created with x maximum health[1] (except muckrakers which are created with $0.7x$). Enlightment Centers can also *bid* influence for votes. The team of the EC that bids the highest on a given round gets that round's vote. Each team begins with between 1 and 3 Enlightment Centers and can acquire more by conquering neutral or enemy ones.

[1]The game uses *influence* and *conviction* to refer to a unit's max and current health respectively, but we will use the standard terminology from now on.

- **Politician:** It is the main offensive unit of the game and also the fastest one, it can attack and hit all units inside a given radius. When doing so, the politician dies, every enemy unit gets hit and every ally unit gets healed by a certain amount. The amount of damage/healing done is the politician's current health divided by the number of units hit (counting allies). If an EC or a Politician's health is reduced to less than zero this way, it gets converted to the attacker's team and the damage overhead is gained as health. This damage can be improved by the *Muckraker Buff*, which is explained later.
- **Slanderers:** During the first 50 turns these units produce an extra income, and after 300 turns they transform into a politician. It is the slowest unit of the game.
- **Muckraker:** It can one-shot slanderers in range. When doing so, it increases the politician's damage by a factor that scales with the killed slanderer's max health.

A team wins whenever all enemy units have been killed or if they have the majority of the votes after a given number of turns.

3 Meta Overview and Bot Development

3.1 Pre-Sprint 1

It was pretty much common knowledge that the initial specs were broken. Whenever a Muckraker killed a Slanderer, all its politicians had an attack factor of 1.01^x during 50 turns, where x was the Slanderer's max health. Not only that, but politicians could attack right after being created, which means that if the buff was large enough (which it was by killing a single slanderer), they could give the adjacent EC more influence than what they were worth by attacking it (this is known as *self-empower*). Altogether, the team who killed the first slanderer could easily make its own influence overflow in a few turns, and win later by turtling and buying all the votes.

Since we knew that specs were probably going to get changed at some point we decided to focus on having good fundamentals and solve problems that we assumed would remain stable for the rest of the competition: **communication** and **pathfinding**.

Communication between units in BC2021 was pretty different from previous editions: each turn, each unit can display a 24-bit integer and all nearby units can read it. Additionally, all units can read the EC integers from anywhere in the map, and the ECs could read the integers of all units. However, in both cases, for a unit a to be able to read the flag of a unit b that a can't see, a needs b's ID.

Pathfinding was also different from previous editions. Usually in Battlecode all tiles are either passable or impassable, however this year all non-occupied tiles were passable but each tile had a *passability factor* that multiplied the additional cooldown of passing units. This multiplier could go all the way from $\times 1$ to $\times 10$.

We implemented a provisional pathfinding algorithm that was a mix of a greedy algorithm between adjacent tiles and bugnav (whenever most of these tiles were occupied) and decided to focus on communication. The reason behind this is that communication was pretty much the backbone of any non-trivial strategy, while pathfinding could be improved at any time with no game-changing effect on the rest of the strategy.

3.1.1 Communication protocol

The high-level idea of our communication protocol is very simple: each unit cycles through all what we considered that was *important information* and displays one at a time. The important information that we considered was the following:

- Boundaries of the map.
- IDs and Team of ECs.
- x-coordinate of ECs.
- y-coordinate of ECs.
- Influence of Neutral ECs.

All of this information could be gathered either by observation or by reading other flags. To store the information about all ECs (and other units) we would use a custom list which had attributes for ID, MapLocation, etc. At the beginning of every turn each unit would attempt to read the messages from all known ECs and remove from the list all ECs that it could not read (this means it probably converted to another team). Information about map coordinates would always be encoded $\bmod 128$ (which is twice the maximum map's size), since this guarantees that all units can reconstruct the actual coordinat value with no error (the closest number to its own coordinate that matches mod 64). Besides this message cycle, units would occasionaly display other types of *emergency messages*:

- Enemy Muckrakers: If an enemy sees an enemy muckraker, it would display its location and influence (capped at 127).
- IDs of other same-team units: Each non-slanderer[2] unit would display the ID of a nearby allied unit only once (this means that the same ID is never displayed

[2]Since slanderers would stay close to the EC this was generally not worth for them to do.

more than once). This way, each of our ECs could gather information from units spawned at other ECs. This allowed our different ECs to be able to read each other's information (since all non-converted units display information about their parents in their cycle).

To speed up communication, we also made units keep track about what information was *known* by their parent EC (if they had one). For a unit u, a piece of information x is known by EC e if u received a message from e containing x. If u has some information x which is not known by its parent e, it would prioritize sending x.

Our communication protocol allowed our units to have almost complete information about the map (or at least about what we cared about the map) in very few turns since units could read information from the EC and all nearby slanderers every turn. At some point in pre-sprint they increased the initial cooldown of politicians and muckrakers to 10. After this spec-change, our politicians and muckrakers would almost always have complete information whenever they could move for the first time.

3.2 Sprint Tournament 1

Before the tournament, **Teh Devs** nerfed Muckraker buff from 1.01^x to 1.001^x, and that made games more stable. In the end, team **Super Cow Powers** took the first place with a pretty solid all-around bot. By the time of the tournament, our bot was still pretty dysfunctional since we spent pretty much the whole week implementing and testing communication. Our politicians and muckrakers would simply go to the closest enemy EC and attemt to capture it (with poor micro), while slanderers would attempt to explore the map avoiding enemy muckrakers. We lost 0 - 3 against team **remotED** in fourth round.

3.3 Between Sprint 1 and Sprint 2

This year **Teh Devs** got rid of the Seeding Tournament and did another Sprint Tournament instead. Personally I'm quite happy with this decision since ladder rankings at the end of the submission period are way more accurate than the results of a tournament one week prior. In previous editions, it sometimes happened that a team that put a lot of effort into getting a good result in Seeding could get matched in the Qualifying Tournament against another top bot who decided to skip Seeding or to improve their bot later on (both teams are being punished this way).

Soon after the first Sprint Tournament ended, we implemented our bot's micro and three very important features that would make our bot go all the way from the very bottom to the number one spot in the ladder (and it would stay there for the rest of the competition!).

3.3.1 Micro

Each politician would compute an *efficiency score* for all possible attack radius, and attack only if the efficiency of doing so is greater than its current health. The efficiency was computed as follows:

- For each ally unit, we add the total health restored.
- For each enemy unit, we add the total health lost (if we suspected that the enemy unit is a politician, this is factored by the enemy buff).
- If a unit is killed or converted, we added a bonus of +17 efficiency (this is how we valued unit advantage). We also added this bonus when damaging an enemy EC.

To guess if an enemy unit was a politician or not, we would take a look at its max health and check if it was one of the slanderer optimal values (basically there were 30 optimal values[3] for slanderers that nearly all top teams used). More precisely:


```
e = 1;
maxInf = 1 + r.getConviction();
switch(r.getInfluence()){
    case 21:
    case 41:
    case 63:
    case 85:
    case 107:
    case 130:
    case 154:
    case 178:
    case 203:
    case 228:
    case 255:
    case 282:
    case 310:
    case 339:
    case 368:
    case 399:
    case 431:
    case 463:
    case 497:
    case 532:
    case 568:
    case 605:
    case 643:
    case 683:
    case 724:
    case 766:
    case 810:
    case 855:
    case 902:
    case 949: break;
    default:
        e = enemyBuff;
        maxInf += r.getInfluence();
        break;
}
```

[3]By the way in which the income formula was designed, building a slanderer with any other influence value would be sub-optimal, since it would cost more but would produce the same amount of influence per turn.

Assuming that other teams built slanderers optimally, the worst case scenario was that we could consider some politicians as slanderers (in fact slanderers do transform into politicians after 300 turns), but usually this was not that bad since with buff they would get targeted and killed anyways. Note that this also allowed us to compute the efficiency more accurately since all overkill damage done to politicians does not go to waste.

We also added a special mechanic to capture neutral and enemy ECs. Our politicians would circle around it (at distance ≥ 5 - this was done implicitly with bugnav since we considered tiles with distance < 5 as impassable) and the strong ones (more than 40 max health) would attempt to go to its adjacent tiles (only the ones at distance 1!). This would guarantee that our units would not obstruct each others when attacking. If the enemy EC attempts to dilute damage by spawning weaker units, these would usually be cleared up automatically by the circling politicians.

The micro for slanderers and muckrakers was straightforward: Slanderers would run away from enemy slanderers (they would prioritize tiles with high passability), and muckrakers would go to the closest enemy slanderer if they have one in sight.

3.3.2 Feature 1: Build Order

Knowing what to build each turn was possibly one of the hardest challenges of Battlecode 2021, the main reason being the fact that you must not only choose the desired unit type, but also the amount of resources invested into its max health. By trial and error we tested several protocols and we ended up doing the following. We would build cyclically Slanderer $\rightarrow$ Muckraker $\rightarrow$ Muckraker $\rightarrow$ Politician. When building a Slanderer we would use all our available influence (of course sticking only to *optimal values*), and when building a Politician we would alternate between small values of influence. When trying to build the first muckraker of the cycle, if we had a minimum amount of influence we would build a big politician instead, using all of our influence except what is needed to build the next slanderer (taking into account our income as well). This build order allowed us to keep our economy in good shape while producing also enough offensive units to put pressure on the opponent.

3.3.3 Feature 2: Muckraker Flanking

We empirically saw that most of the muckraker potential was wasted by only going to the closest enemy EC. These scenarios were particularly bad:

- The opponent is close to our starting point. In this case, muckrakers would go directly to the opponent and ignore the rest of the map, which takes a toll on exploration.

- Contested middle ECs: Whenever there are neutral ECs in between, our muckrakers would focus on these (if they are being constantly recaptured by our opponent) while ignoring the opponent slanderers, which are usually concentrated in the starting ECs.

We partially avoided these issues by having muckrakers go in each direction and follow the borders of the map for 150 turns. After this, they would go to the closest EC as usual.

3.3.4 Feature 3: Slanderer Protection

Probably the most obvious fix to our bot was having Slanderers stay near the closest EC (we place them following a lattice, inspired by team **Producing Perfection**). This way they would have zero cooldown and would move immediately after seeing an enemy muckraker, and also they would be surrounded by new units from the EC, which means that they would have more time to react if an enemy muckraker is nearby since this muckraker would probably be spotted by some of these new units before it could get close to the slanderers. Additionally, the EC would spawn politicians if there are enemy muckrakers nearby (because of the notifications from nearby units, the EC could prepare a politician even before the muckraker enters vision range!).

Our implementation had an unexpected feature, which is that if we have several ECs, our slanderers would occasionally move to the safest EC. This happens because if an enemy muckraker herds them away from their original EC, the *closest* EC might change and they might start settling around a new one.

3.3.5 Non-feature: self-Empowering v2

During the last days before Sprint 2, many of the bots started to implement a new method of self-empowering: whenever they would get a significant buff (~ 2), they would spend all their resources into creating big politicians and empowering their own EC every 10 turns, which is the number of turns required to empower after being built. With the influence obtained this way, they would create huge muckrakers impossible to take down, and would then get an even larger buff (which they would use to get even more influence). In the end, they would just turtle and win by votes. The most notable teams performing this strategy were **Nikola**, **Chop Suey (California Roll)**, **Confused**, **Blue Dragon** and **Kryptonite**, although nearly all teams self-empowered one way or another.

The only reason we didn't implement self-empower was because we were pretty sure that **Teh Devs** were going to nerf it at some point. We thought this because self-empowering was not just an almost self-sufficient easy-to-implement strategy (in the past there were many strategies such as rushes / turtles that were never nerfed), but

it was also boring to watch, made the games way too random (since you can pretty much guarantee a win by killing a couple of enemy slanderers), and most importantly it was a strategy that you could implement on top of any bot, so in the end, regardless of how you get the buff, all bots would be forced to self-empower once they get it if they wanted to stay competitive, since it was clearly the most robust and efficient way to use your buff.

3.3.6 Non-feature: Politician Lattice

Most of the teams placed politicians in a lattice around the EC (and the slanderers) to protect slanderers from enemy muckrakers, most notably **Producing Perfection**, **Chop Suey (California Roll)** and **babyducks** (although their politicians were not in a fixed positions). In our local tests, we considered that having some of our politicians stay near base was wasting too much offensive potential to be worth, so in the end this was not implemented and we relied only on communication + politician spawning from EC for defense.

3.4 Sprint Tournament 2

Even though our bot did pretty well in the scrim server, we lost in the 1/16th round against **Kryptonite** in the Sprint Tournament 2. The large and mostly impassable maps in that round made defending against enemy muckrakers really hard because of unit congestion (barely any politician attack is worth when the whole map is filled with units!). And in the end our decision of not implementing self-empowering turned out to be critical since **Kryptonite**'s bot was able to use its buff much more efficiently than ours. In the end Sprint Tournament 2 was won again by **Super Cow Powers**, who is the first team to win two Sprint Tournaments in the same edition! :)

3.5 Post Sprint 2

A few days after sprint 2 **Teh Devs** nerfed self-empowering and reduced buff to $1 + 0.001x$, however it didn't affect the rankings too much since generally all top self-empowering bots were good even without self-empowering. In particular, team **wololo** got to the top 10 page by doing a Muckraker rush and spawning almost no Slanderers or Politicians. During this period we added two main features:

3.5.1 "Good" Pathfinding

We decided it was time to replace our greedy algorithm by a stronger one, and in the end we chose to perform a weak version of Dijkstra, in which we only consider

paths that move away from the origin in every step. The advantage of doing so is that we can compute the distance to all visible points using a simple recurrence: the distance to a tile t is t's multiplier plus the minimum distance of all tiles adjacent to t that are closer to the origin. Since the comparisons that the algorithm should make are already predetermined given the unit's vision range, we could do all necessary comparisons without any *for* or *while* loops. In fact, we generated the necessary code for this procedure using another program, and we ended up having a 3000-line method with 400 if-statements. However, it only took 6000 bytecode to check all paths in the Muckraker's vision range!

3.5.2 Improved Build Order

Our build order used for Sprint 2 was surprisingly hard to improve, in the sense that we couldn't get a considerable winrate against our old bot by only changing build order. In the end we substituted our fixed build order by a more flexible one by giving *weights* to each unit and we would always try to build the unit with the lower total weight. Whenever that unit is build we added that weight to its total weight. The advantage of building units this way is that we can change the weight over time and easily prioritize some units over others: for instance we would focus less on slanderers and muckrakers mid-game and build more politicians.

A second improvement consisted on creating our own enum with custom types such as *Big Politician* or *Big Muckraker*. This way it was easier to control when we are building each type of unit.

We also made our ECs produce *conqueror* politicians to capture Neutral ECs as soon as they could (as long as its cost isn't more than twice of the cost of the highest slanderer built). We build these politicians with health equal to the neutral EC's health + 10.

A final "improvement" was made in which we controlled the growth of our economy by having that after building a slanderer with the kth optimal value (out of the 30 possible ones), the next one could not be larger than the $(k+2)$th. We realized that our economy was slower in the end game, but in the early and mid game we would get an offensive advantage that could capitalize way more than investing into economy would.

After performing these changes we increased our elo advantage in the ladder and had no trouble making it into the Final Tournament.

3.6 Pre-Finals

Before finals, we had a couple of days to further improve our bot. From what we saw in the scrim server most of the other bots (**Chicken**, **Chop Suey (California Roll)**, **babyducks**, **monky**, etc.) were leaning towards a heavier economy build, however we did several local tests and we didn't find building a larger economy to be that helpful since it made you more vulnerable in early/mid game. We found finding the sweetspot between economy and offense really hard since it was heavily map dependent (economy being better the larger the map), so we basically chose the one (out of the ones we tried) that had the most winrate against our old bots on the maps used for Sprint and Sprint 2. During this period we also added the feature that large and middle muckrakers would not go to explore or go around the borders of the map but would hop from enemy EC to enemy EC (if they see one they go to the next one). This would guarantee that they would not spend too much time on an enemy EC with no slanderers (if they see slanderers they still prioritize those to anything else).

3.7 Final Tournament

Relying on previous maps turned our to be a pretty bad idea since from the very first rounds we saw that final maps were huge and/or with lots of highly impassably tiles. I suspect that most of the teams saw this coming since most of the smaller maps came from the Sprint 1 tournament, while the Sprint 2 and Qualifying maps were larger and less passable on average. The effective distance between us and our opponent made that our bot could not take advantage of a heavier midgame military force since whenever our politicians and muckrakers reached the opponent, they would have already built strong defenses. We lost in winner semi-finals against **Producing Perfection**, and later in loser semi-finals against **babyducks**, who had an impressive run in the losers bracket all the way from the very first game to winning the whole tournament! Our bot fought hard until the very end, having lost both games by 2 - 3.

Overall we are quite happy with the results and with the fact that our bot kept its number 1 spot on the ladder for the entirety of the second half of the competition.

4 Final thoughts

I really liked this year's Battlecode overall. As usual, **Teh Devs** brought us a quite original game with unique features, mechanics and several viable strategies. I'd like to thank **Teh Devs** for putting all the effort required to run a competition as large and complex as Battlecode, and also to all sponsors who support the event even though

it was going to be remote.

The only downsides I can find about this year's edition is that games ran slow (although nothing compared to the non-Java engines!) and that specs were kind of broken until pretty late into the competition. Honestly, the specs issue is not a problem for Battlecode veterans since we know that most of the bugs/exploits get patched eventually and we can plan ahead, but it might be frustrating for newcomers to have spec changes so late (in this case they were definitely needed, but it would have been helpful to have them earlier).

Unfortunately this year we couldn't have live finals because of Covid. Meeting other participants and **Teh Devs** is always the best part of Battlecode (sponsors usually send pretty cool people too! Some of them are ex-battlecode devs/participants). Despite the inconveniences (and time-zones!), Battlecode has a strong community and I had a lot of fun during the virtual dinner and the Final Tournament stream. Hopefully, there will be no restrictions for 2022. Although this year I couldn't ~~force~~ ask other participants to play board games, I'm always up in Discord for some remote Terraforming Mars, Pokemon Showdown or League :)