CMPUT 350 Report

While working on Cyanide there were many goals we wanted to achieve. Our main focus was to make changes that would increase the capabilities of UAlbertaBot for all of the races. For this reason a large portion of our effort was towards very general changes. Another goal was to increase the variety of strategies for the Terran race, and make improvements to existing ones. We originally wanted to implement a Support Unit role, a Siege Tank Push tactic, and a Dropship role. Unfortunately, due to time constraints, the Siege Push tactic was only partially implemented, and we were not able to get Dropships working due to bugs in the existing code. For this reason we have included a simpler Siege Tank implementation, and no Dropship implementation.

The reason behind an introduction of a Support Unit Manager is that this allowed us both to increase the capabilities of UAlbertaBot in general. But also gave us an easy way to implement the Medic Support Role. Thus it fulfilled two of our goals. Our implementation of the Support Unit Manager was designed to be very general. We wanted a design that could be used by any race and any support unit. In addition we wanted the support role to be usable by both offensive and defensive units.

The solution we came up with was the class SupportManager, which subclassed MicroManager. This class plays the same role as MeleeManager and RangedManager but for support units. The difference is that this class uses SpecificUnitManager to implement the support unit behaviour. The SpecificUnitManager class can be used by any unit type, but to have more specific implementations, subclasses should be made such is the case for Medics and Comsat Stations.

The default behaviour of the SpecificUnitManager allows units to defend nearby allies. We used this as a default implementation because it is a role that almost any unit can perform. Another reason for this default behaviour was that in our Siege Tank Push we wanted to have marines defending Siege Tanks from flying units. This would allow us to multi-purpose the support role.

For specific support unit implementations there are two major methods that are used by the SupportManager. The first is getBestAction(). This method takes in a list of allies, and a list of enemies. It is used to get the best action for the support unit. An action consists of an order and a target.

In the case of the Medics, getBestAction() either returns the Heal order with the most damaged organic ally, or the follow order with the nearest attack unit if there is none to Heal.

This behaviour is used to heal allies most in need of healing. And stay close to allies that may need healing soon.

In the case of the Comsat Station, getBestAction() returns the Scan order with an undetected enemy near our units if we are attacking and near one, or no action if there is no undetected unit near our attack point. The implementation of the Comsat Station was a demonstration of how extendable our SupportUnit implementation is. Originally it looked like a lot of changes were going to be needed to add to get Comsat Station Scanning to work. But we quickly realized that the SupportUnit role was a perfect fit for the Comsat Station. The result was a very short and elegant implementation that supports multiple Comsat Stations, and is conservative with energy use.

While we did not complete the Siege Tank Push like we wanted to, we did implement a lot of support for Siege Tanks. There were a lot of issues that needed to be handled before we could even add the micromanagement for Siege Tanks.

First of all, we had to add support for building add-ons. Siege Tanks require that each factory that produces them has an attached Machine Shop. The add-ons would not build originally because the BuildingManager could not figure out how to build them. This was a relatively easy fix, however this led to a host of other problems.

The first problem we faced was that Siege Tanks could only be produced by Factories with Machine Shops. The result was the build order queue trying to build Siege tanks from a Factory without a Machine Shop, and the build order queue stalling. The solution was to change the ProductionManager so that it would only choose buildings with an add-on when building units that require that buildings add-on.

The next problem we faced was that a Machine Shop was required at each Factory that we wanted to produce Siege Tanks at. But the BuildOrderSearch would only build one Machine Shop. The proper solution would require modifying the BuildOrderSearch to include this requirement. However that code was going to be quite hard to modify and maintain correctness. There was also the possibility of manually inserting Machine Shops into the Build Order, but this seemed quite hacky. The final solution we came up with was to request as many Machine Shops as there were Factories in the StrategyManager. A more general fix would be nice, but this worked for our use case, and saved us a significant amount of time.

The final problem we faced again had to do with add-ons. Despite the fact that we had add-ons building, and Siege Tanks only being produced at Factories that could build them, there was one more problem to deal with add-ons. There were times when a building would be built

too close to a Factory, and as a result the Machine Shop could not be built. This results in the build order stalling when the Machine Shop can not be built, and we do not build any Siege Tanks. We added some code to building placement to prevent placing buildings within the tiles that a potential add-on can be built. This allowed us to guarantee that the add-ons could be built any time after the base building was built.

After all of this effort we had Siege Tanks that could go into Siege Mode. At this point we put some effort in to get a Siege Push working. However, after some time it was decided that the effort required to get a good Siege Push could be better invested in working on our other strategies. We decided to only use Siege Tanks in a defensive manner. As such code was added that gets Siege Tanks to go into Siege Mode near the entrance to our base.

Since we abandoned the micromanagement needed for a proper Siege Push, we had a lot more time to invest in our other strategies. There were two types strategies we needed at this point. One was a rush strategy to beat bots that do not build up a defense quickly enough. For this we chose a Marines and Medics strategy since it is slightly better than a simple Marine rush due to the Medics increasing the lifespan of Marines. In addition we could take advantage of the Support Unit class we had written. The second was a defensive strategy to hold out against bots that have a better rush than we do. This mainly applies to Protoss since Marines and Medics are not able to contend with a Zealot rush.

Our defensive strategy consisted of building Bunkers at the front of our base, and filling them with Marines. In addition, around mid game, the Bunkers are supplemented with Missile Turrets. The reason we chose this tactic is that a single Bunker with Marines can easily pick off small packs of Zealots, and is easy to construct quickly. Multiple Bunkers can hold off many Zealots, and we can get them quite quickly, well before Protoss can swarm our base. We supplement the Bunkers with Missile Turrets to prevent Dark Templar rushes, and cloaked units in general.

We noticed that sometimes after a few waves of Zealots a Bunker would be destroyed. We were concerned that after many waves we would lose all of the Bunkers. As such we added some code that gets idle workers to repair damaged Bunkers and Missile Turrets. This made a huge impact on the life of Bunkers. Making it much harder for Zealots to breach our base.

The defensive aspect of this strategy works great against early game rushes and can hold out for a fairly long time. However, we still need an offensive part for the strategy to win games. The original plan was to hold out until mid to late game and either nuke our enemies mineral line or dropship some Marines and Medics into the back of our opponents base to kill their economy. We found this to be a very effective strategy when we tried it manually against

bots. We were going to accomplish this by supplementing our Bunkers with Siege Tanks in Siege Mode, allowing us to rush to Dropships and Nukes, without threat of enemies getting into our base.

The problem we faced when trying to get Dropships and Nukes is that StarcraftData has a list of units used by BuildOrderSearch, and it seems to only use the first 30 elements of the list. This means that we can only build 30 different types of units, upgrades, and technologies. We found one of the lines of code causing this and fixed it, but the problem persisted. We were unable to find the other bug causing this behaviour. For this reason, we are unable to build late game units since we can not fit them in the first 30 elements. As a result we decided to change our strategy.

Instead of getting Dropships and Nukes we decided to use Marines since we already had them working. So our defensive strategy consists of building up some defenses and then continuing in a normal rush strategy. This does not sound like it would be effective but it turns out that in the case of Zealot rushes it is quite effective due to the Zealots being picked off by the Bunkers, and never building up a large force.

One final change that we wish we did not have to make but increased our defensive strategies win rate was the removal of Siege Tanks. Because we were unable to make it to late game and were rushing, it worked out better in most cases to get Marines as quickly as possible to harrass our opponent. Taking the time to get Siege Tanks sometimes resulted in a loss due to our opponent getting too many mid-game attack units to our base.

Overall our changes resulted in a very successful Terran bot. We summarize the results of a tournament we ran in Figure 1. As can be seen, Cyanide places in 4th, with a win rate of 52.59%. This more than meets our best wishes for the improvements we made. We did not expect to beat either UAlbertaBot or Skynet in the competition, and we had low hopes for beating any Protoss bots in general due to their strength.

The tournament results revealed some interesting facts about our bot. And watching replays gives us insight into the improvements that can be made. From the tournament results we can see that most of the matches that we lose are to UAlbertaBot, Skynet, and Aiur. Aiur's wins are the most interesting of these.

Aiur beats Cyanide by using a Photon Cannon rush, placing Photon Cannons within our base and picking off our workers. This strategy beats us because a Photon Cannon can be made before we can defend ourselves. One improvement that may solve this problem would be to have a scout worker scouting out our base and attacking any Probes that enter. The current

code attacks Probes in sight, but if they stay in the fog of war then we will not attack them and we will lose.

	Bot vs. Bot Results - (Row,Col) = Row Wins vs. Col									
	Win %	UAlbe	Skyne	Xelna	Cyani	Aiur	Ximp	ICESt	Nova	BTHAI
UAlbertaBot	89.29	-	12	12	11	13	14	14	10	14
Skynet	71.05	2	-	9	14	12	12	9	8	15
Xelnaga	58.04	2	5	-	6	10	7	10	13	12
Cyanide	52.59	3	0	8	-	5	9	13	11	12
Aiur	51.28	1	3	4	10	-	6	9	12	15
Ximp	45.54	0	2	7	5	8	-	9	7	13
ICEStarCraft	43.97	0	5	4	2	6	5	-	15	14
Nova	34.48	4	6	1	4	3	7	0	-	15
BTHAI	5.98	0	0	2	3	0	1	1	0	-

Figure 1: Mock tournament results with AIIDE 2013 entrants and Cyanide.

In addition, we would be able to beat UAlbertaBot a lot more often by using late game units. This is the case because we are often able to hold them off with Bunkers, but our Marines are often not enough to push the Zealots back to our opponent's base. Because the Zealots harass us so much, we never expand, and because UAlbertaBot is able to expand, it is able to wear us down over time until we are out of minerals and lose. Thus the addition of late game units would allow us to push back harder and hopefully win. Unfortunately as mentioned above there are bugs in StarcraftData that prevented us from getting late game units. As such, the next thing we would like to fix in Cyanide is the ability to get to late game units.

In addition to these opponent specific changes, there are some general improvements we would like to make. First we would like to improve the Siege Tank micromanagement, so that it can actually be used to perform a Siege Push. We believe that with the right defense and proper micromanagement this strategy would be able to win against most bots, the exception being bots that invest heavily in flying units.

A major change we would like to implement is an in game adaptive strategy. There are some bots that attempt this but there are no great mechanisms to do this yet. The reason we think this change is essential is that a good bot needs to adapt to changes as they happen. If it does not then human players will be able to anticipate its actions and counter them. With adaptive strategies, we would be able to determine the attributes that make up our opponent's army and produce units that will counter that army.

The final change we would like to make is an improvement to the kiting used by ranged

units. The existing kiting seems to get our ranged units attacked too much, due to them running back and forth too much. In particular we feel that the use of Vultures with proper kiting would be a good early game strategy to supplement our Marines and Medics strategy.

In conclusion, we are quite happy with the result of our efforts. We managed to make a Terran bot that greatly surpassed the AIIDE 2013 Terran bots. In addition we managed to make general improvements to UAlbertaBot in the form of the generic SupportManager class hierarchy. We believe that these classes would be very valuable to all races for use by spellcaster units or those units needing defense from certain kinds of enemies. Given more time there are many improvements that we would like to make, but given the time constraints, we believe we made a large impact on the existing Terran bot.