Developing a StarCraft: Brood War agent

Carsten Nielsen

Kongens Lyngby 2015

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

The goal of the thesis is to ...

Summary (Danish)

Målet for denne afhandling er at ...

Preface

This thesis was prepared at DTU Compute in fulfilment of the requirements for acquiring an M.Sc. in Engineering.

The thesis deals with ...

The thesis consists of ...

Lyngby, 30-June-2015

Carsten Nielsen

Acknowledgements

I would like to thank my....

Contents

Chapter 1

Introduction

Goals, challenges, programming language.

Other projects and agents.

1.1 The "separate document"

Original project plan.

Revised project plan.

Brief self-evaluation.

1.2 Starcraft: Brood War

Game description: premise, goals, races, units, challenges.

For this bot I focused on Protoss. The main AI challenges between races are shared, however specific gameplay elements differ quite a lot.

1.3 BWAPI

BWAPI description.

BWTA and SparCraft description

1.4 AI Competitions

Competition rules: Setup, limits, goals.

1.5 Process

Chapter 2

Datastructure Design

2.1 Overall Design Paradigm

Modular.

Hierarchy (Limits, benefits).

Modules and Managers. Managers are modules and are updated every frame. Non-managers are not autocratic and is used purely through function calls, through events or other modules.

Other designs: Single manager (More human-like)

Chapter 3

Information Management

Intro

3.1 Scouting

Scouting is important throughout the entire duration of the game. Early on, players need to know where the opponent is, what faction they are playing and which strategy they are employing. Later on, observing the enemy army size, unit types, base expansions and tech level is important to counter strategies.

Enemy opening -> opponent modeling.

The opponents faction is important when deciding on opening strategies. The factions behave very different early in the game. Openings are a decisively slower when one has to defend against every possible attack. Only if the opponent picked random as a faction will it be hidden. No units are shared across factions, so the first unit discovered will reveal it, which usually is their scout or main-base.

Base positions + expansions

3.1.1 Initial Scouting

The map terrain, along with resource locations, are completely revealed at match start. This includes all possible player start locations. Scouting for the opponent base is then just going through all other start locations. Maps used in competitive matches are usually no larger than four-player size, meaning there are four possible start locations. A player then has to take into account that they might have to scout up to two bases before knowing the enemy start location. Usually a player wants to know more than where the opponent starts, so even when it is deductible where the opponent is, the player wants to scout the base itself.

Enemy opening.

Once the enemy base has been revealed, and with it his opening strategy, the scout is not useful any longer. It is a long trip back to gathering resources, and the scout might be followed, revealing your own location. Harassing enemy workers can put a dent in the enemy economy, even if none of them die. Simply by attacking enemy workers forces the opponent to pull two from gathering to defending. From there, the harassing scout can retreat until it is no longer chased, or lead chasing workers around while attacking passive gatherers. If the scout manages to kill a worker the opponent will fall behind in economy. However, compared to competitive human players, bots are usually too inefficient to take full advantage of this, but every bit helps.

There are additional uses to the scout however compared to harassing. These must all be considered in the grand strategy, as they involve spending resources. These strategies include proxy bunker, photon cannon, barracks or gateway, manner pylons or gas stealing. Proxy structures involve building right below the enemy ramp or even inside their base. The usual distinction here is whether the structures keep the opponent in with defensive towers or rush attack with front line troop producers. Manner pylons are used in conjunction with these as Protoss, where the required pylon for forward bases are placed within the enemy mineral line, blocking and possibly caging enemy workers. Gas stealing involves building a refinery on the enemy geyser, blocking gas harvesting and forcing tier one unit use. While proxy troop production has not seen much use in bots, both gas steal and proxy towers has been used for varying effect.

3.1.1.1 Implementation

Usually build orders include specifically when to send out a scout. In case of Protoss, the scout is often a worker that has just warped in a structure. This would however require a build-order system capable of containing other elements than builds. A simpler solution which is used here is sending a scout when a specific supply limit has been reached. This is noted in build orders, allowing for only slightly inaccurate build-order implementations.

When picking a scout, the bot searches through different worker groups. First the idle, then the mineral miners and finally the gas harvesters. When it comes across a worker not currently carrying any resources, it assigns it as a scout, removing it from the Task Master in the mean time. Contrary to building or defending, scouts are assumed to not return, so it can safely be removed from the local worker pool and harvesting. If the scouting manager is unsuccessful in finding a worker, it retries next frame.

While we have a scout and do not know the opponents position, we pick an unexplored base location and move the scout there. A tile is unexplored if it has never been revealed for the duration of the game. Thus, our home base will not be considered for scouting, and once the scout reaches the destination the tile will be explored, removing it from the possible scouting locations. Exploration is handled by BWAPI.

This implementation scouts as long as no enemy buildings are known, extending its use into late game. Usually fast or cloaked units are used to scout later in the game, but it is not necessary.

3.1.2 Intelligence Management

Locating positions, types.

Geysers <-> refineries

Possible Additions: Faction

Chapter 4

Resources and Bases

Intro

Task Master

4.1 Resource Management

In early- and mid-game, *economic advantage* is the most important aspect of StarCraft. This is when you have a higher resource input than your opponent, and happens when you focus more on resources than them, or they sacrifice long term advantage over short term. The point of this is if you eclipse your opponent in resources, you can perform worse in combat and still end up on top. You can send more units, recover them faster and expand economy quicker, retaining your advantage.

There are three resources in StarCraft. Minerals are mined from mineral fields which usually are in clusters of 8-12 by each base location. Everything you produce costs minerals, so it is the most important resource. They are harvested by your workers, which bring it back to a nearby resource depot in batches of six. Usually there are 2-3 workers on each field depending on the distance from

the depot. *Vespene Gas* or simply gas is harvested from *refineries*. These must be built upon *Vespene Geysers* of which there are one or two by placed by each base location. It is harvested in batches of eight with a max of three workers per refinery at optimal depot distance. Advanced structures, units and all technologies depend on gas, and a decision with major impact is when a player starts harvesting gas. The immediate cost of the refinery and lost mineral gathering is a liability against fast rushing strategies. The last resource, supply, is reserved by units. Every unit reserves some amount of supply which is released upon their destruction. The only way to secure more is building *supply depots*, *pylons* or *overlords* respectively for Terran, Protoss and Zerg. Each of these add some amount of supply, which is removed if they are destroyed. A player can happen to use more supply than they have, but will be unable to build more units until more supply is acquired.

Only one worker can be active at a mineral field or refinery.

Minerals, gas and supply.

StarCraft has a built in worker gathering AI. Workers will automatically return cargo from resources (unless they are interrupted). When gathering minerals, the workers will move to another mineral field if the current one is occupied. When gathering gas, they will wait until the refinery is unoccupied and then gather.

Mining Minerals

The simplest mineral gathering implementation is ordering idle gatherers to mine some arbitrary mineral. At some point, the built-in AI will ensure the workers are optimally scattered. It will however not be scattered immediately, and some workers will be very inefficient while moving from mineral to mineral. A simple but effective addition would be to scatter the initial workers.

By maintaining a queue of minerals, we can optimally scatter the workers. The first element of the queue is the mineral with fewest workers and the one in the back has the most. By continually assigning new workers to the first element and moving it to the back, we maintain a queue where the last mineral has at most one more worker than the first. Removing workers however requires finding the mineral in the queue, and should be done sparingly. Therefore, it is assumed any building or defending activity will be short and temporary, and workers assigned such will not be removed from the scattering. This might result in an ineffective scattering at some points. It is not clear however if optimizing the scattering at all times results in optimal resource output, as workers might

be moved between minerals too often, resulting in less time mining.

If we maintain a dictionary of workers and their targets, we can assign new ones in constant time and retrieving targets in logarithmic time. This could be improved to amortized constant time with hashing. Removing a worker however requires a search through the queue which is linear time.

Other Implementations.

Harvesting Gas

Implementation.

Multiple refineries.

Building Supply

A player can have a maximum of 200 supply. The simplest supply implementation is to build supply units whenever the supply limit has been reached while it is below the maximum threshold. This will however throttle production while the supply structure is being constructed, which is inefficient. The optimal solution would be predicting supply needed in the near future and interlace supply production with unit production such that all orders are completed the earliest. Observe however that this is alike the *Job shop scheduling* problem but more complex with the additional factor of resources. As such the problem is NP-complete, and even then resource gathering rates must be predicted for optimal solutions.

A simple solution which is implemented in the current bot is to build supply units when released supply is below a set threshold. This threshold is somewhat arbitrarily set according to testing. A more dynamic solution would be to set the threshold to the amount of supply used if all production facilities built a unit, assuming the unit type can be predicted. This solution will build supply units that are completed before they are needed, but might not result the optimal amount of units completed at any given time (impacting economy with relation to workers). The worst problem, production throttling, is avoided however.

Managing gathering

Usually a Protoss player should not stop building workers until late game, but some openings require a temporary pause. Therefore the solution is just greedily building workers while no build order is being executed.

Two workers per mineral is the optimal amount. The third worker will have diminishing returns and the fourth would have no impact. A third worker is optimal if the mineral is far away from the depot. Three workers on every refinery is the standard, regulated only according to need. Usually a refinery will be at max capacity, unless the player has lost too many workers.

Mineral to gas ratio.

Multiple bases.

The Production chapter will cover how and where we build these expansions.

Maynard Slide

Chapter 5

Production and Expansion

A *Unit* is any player controlled entity, building or not. Some spell effects, like the Terran nuke, are oddly enough also classified as units but they are neither selectable nor controllable. This is allegedly because of time constraints, and when the term unit is used it usually does not include these. In this paper we sometimes refer to non-buildings as units, which should be clear from context. Buildings are sometimes called structures, usually to avoid confusion between building as a verb and a noun.

Production here refers to the player controlled act of creating a new unit. Officially producing a building is called *building*; *training* if it is a non-building. The term production has been devised for this paper to refer both building and training, whichever is appropriate. Note that the act of building is creating a unit while building a structure usually refers to the entire process of making structures.

When a production has been ordered, there is a duration where the unit is *constructing* before being complete and usable. The event of a unit being created and starting construction is separate from ending it and being completed. An incomplete unit cannot execute any commands and a player will not receive any benefits like additional supply before completion. Buildings are placed in the world prior to construction while non-buildings are hidden, appearing at their constructor upon completion. A unit consumes resources upon production,

including supply requirements.

Expansions are additional depots beyond the starting one, built at other resource clusters. One of the most important concepts in StarCraft, expansions allow players to harvest more resources at higher rates than with only one base. Every tournament legal map has a specific expansion called the *natural expansion*. This is the closest base location to a players start and is usually easily defended compared to other expansions. It usually has less resources than other expansions to compensate.

5.1 Training Units

Producing non-building units is simple compared to building structures. The act of producing a unit is called *training*, however confusingly an incomplete unit is considered *constructing* as if it were a building. The steps can be described as firstly acquiring a relevant and available facility, and secondly monitoring the training.

Implementation

StarCraft has a simple system of who-constructs-who, as there only exists one possible constructor for each unit. All buildings refer to the relevant worker and all units refer to at most one structure. BWAPI has already implemented this functionality, so given an arbitrary unit type we can easily determine if it is a non-building, and if so who trains it.

The agents manager for training and monitoring units is called the *Recruiter*. It is very low in the module dependency hierarchy, while accessed by many other managers. It is not an independent AI, updated only through events. It contains all units capable of training other units. The implementation is generic enough to allow training any train-able unit in the game, even those trained by non-buildings.

5.2 Building Structures

Like training units, building structures is in two stages. The structure is built and then it is constructed. Contrary to training, structures require workers to

be built, where the worker must first arrive at the desired structure location. All player structures are built by one of the three workers (except Terran extensions, but they are handled more like upgrades).

To build a structure we must first find proper placement, find an available builder and then monitor the structure until completion.

Structure Placement

There can be a lot of strategy behind the placement of buildings, and in special cases it is imperative.

Terran can build structures anywhere on the map. Zerg can only build on *creep* which terrain that has been transformed by specific nearby Zerg structures. Protoss can build the Nexus (depot) and Pylon (supply structure) anywhere on the map like Terran. All other structures must be *powered*, which is satisfied when they are within a certain distance of a pylon. A structure can lose power if all pylons nearby are destroyed, and will become inactive until new pylons are constructed.

The pylons are therefore important when placing new structures. The Protoss player must place at least one pylon in any area he wishes to build in and should distance pylons in a base to allow more structures. Fortunately, Protoss bases are usually compact enough such that one can place pylons arbitrarily and still manage to fit other structures in the region. It is important however to place at least one pylon in an expansion before other structures can be built.

Unless a specific building location is specified, the agent attempts to place the structure as close as possible to the main depot of the desired region. This is easily implemented, avoids some pitfalls and accomplishes some goals. All the workers in the region are usually at the mineral fields by the depot, and therefore will have a short distance to travel. Additionally the workers and depot are shielded since they are the furthest point in the base from all angles. The opponents' troops are forced to either destroy the other structures first or move through the bottlenecks of the base.

Even if a region only has one exit, it might be desirable to spread out all structures, as flying units can attack from any angle. Depots will always be placed in new regions at the location BWTA has marked. Refineries can only be built on-top of vespene geysers and are exempt as well.

Implementation

The implementation to place buildings is simple but expensive. The agent attempts all locations in the map in a spiral pattern around the depot, and returns the first location that is available. There are multiple factors when deciding availability. Obviously there must not be any units in the area and the terrain must allow the structure placement.

Placing buildings too close can block passage between them, especially for larger units. The structure hitbox is somewhat arbitrary in StarCraft, as different combinations of buildings next to each other will allow different units to pass through, even though it is tile based. This is because the structure hitbox usually does not cover its location completely. By placing buildings at least one tile from each other, we ensure all units can easily pass through. The agent does this by keeping a tile map of all structure locations inflated by one tile in all directions. If a build location overlaps any of the occupied tiles it is not available. Registering new buildings and querying this is constant time operations, but the space used is polynomial to the map size. It could be improved by only keeping a map for regions we have bases in, although not an asymptotic improvement.

We would also like to avoid placing structures between resources and depots to avoid blocking workers. This is simply done by drawing the smallest rectangle including the depot location, geyser(s) and mineral fields, and avoiding placing structures within this. Building the rectangle is linear to the amount of items in it and querying it is constant, but we expect the amount of items to be low (ten or less).

Aquiring and Commanding Builders

The naive solution in getting a builder is picking one arbitrarily from all controlled workers. This however might interrupt other jobs, pick a worker that is carrying resources, or pick someone far away from the build location.

Instead, recall the `TaskMaster` module from resource gathering. Since all our workers are separated in base locations, we can easily pick a worker from the closest base location. Usually our buildings are placed close to a base location, and in case of expansions we pick a worker from the nearest base. Additionally, the task master marks the jobs of all workers, so to avoid interrupting other jobs it picks a worker from idle or gathering groups (in that order). Finally, the agent searches through the groups until it finds a worker that is not carrying resources. This becomes the builder.

The operation time is linear to the amount of workers, as they are usually gathering resources. However it is probable that less than half the workers are returning resources at any given moment, so it is expected that we only need to check two workers before finding a viable candidate.

Finally, we order the worker to build the construction. A player cannot build in fog of war, therefore the worker should first be moved closer to the target location. When it has been revealed the builder is commanded to build the structure.

Monitoring Construction and Completion

While the structure is constructing but not finished, we need to store it somewhere in the agent. The agent would need to know which structures is soon available, otherwise the events which spurred the construction of the structures might be repeated. There is a lot of frames between scheduling the structure and actually receiving it, where unexpected problems could occur.

If at any point the structure is unable to be completed, it is desirable to cancel the order rather than repairing it. This is because the unexpected event which canceled the structure might have changed the overall strategy and remove the need for the structure. This way, the meta strategy of when to build structures are moved upwards in the hierarchy, reducing independent AI in this module and making it more accessible by other modules.

Cancellation of a build order might occur if for example the builder is destroyed or the build location is invalid.

Implementation

Both Zerg and Protoss structures auto-construct while Terran require a worker. So the agent only needs to keep a dictionary of all structures that are constructing. While the structure has not yet been built, it can only be identified by its type as no unit exist yet. This implies the need of a multi-set, assuming we want to be able to build multiple structures of the same type simultaneously.

The current implementation spends logarithmic time to query, insert and remove new build orders and constructions, but this could be improved to constant with hashing. Both containers are linear in size. Usually a player is not building very many structures at the same time so these improvements are not important.

To ensure tasked workers are building the required structures, the data structure has to be updated every frame. Units have a bad habit of canceling commands in some cases and workers especially do this by retreating from combat. It might also be the case that a worker was given a command in the same frame it was tasked to build, in which case it could not receive the new build command yet.

Since refineries are built on top of geysers, they never trigger the event `create` but `morph` instead. Additionally, when they are "completed", no event is triggered. Completion checks are required every frame to monitor the construction of refineries.

The building, construction and monitoring of structures is handled by the module `Architect`, which also include acquiring workers and placing buildings. The Architect is an independent manager, since it has to command the workers. It has carefully been low in dependencies, to keep it low in the module hierarchy. The building map used to avoid collisions and preserve distancing is contained in the `BaseManager` module. The `Accountant` module is updated with the current schedule of structures by the `Architect` which is queried by other modules. Monitoring refineries is done in a separate module called `Morpher` which handles all morphing units.

5.3 Building Expansions

As noted in the resource chapter, it is often profitable and usually necessary to expand resource harvesting to new locations. To avoid transporting cargo all the way between regions, players have to build resource depots near resources. If the internal data structures behind building structures is limited to regions it may be challenging, and it is in our case.

To build an expansion we need the location and the resources.

Locating Expansion

There are a few things to take into account when expanding. Obviously we need to expand into a location which is not occupied by enemy forces, especially not if they already have a depot at the desired location. Additional factors are proximity to existing bases, defensibility and resource quantity. Expanding to the natural expansion is usually always the first choice, as it scores high marks

on both proximity and defensibility. Usually, the richest base location is in the center and is difficult to defend.

When the agent looks for the next expansion location, it recursively searches neighboring regions from the starting one. The first region which is not already occupied is picked.

Building Depots

When BWTA analyzes a map, it marks optimal base locations, where a depots distance to a mineral cluster and geysers are minimized. Usually each region only contains one base location, at least this is the case for all the AI tournament maps.

Building an expansion is then handled by ordering the depot construction at the found base location with a worker from a nearby region. Currently, the agent always picks a worker from the main base, which is not always optimal. A better solution would be recursively searching neighbor regions' base locations in a priority queue, visiting shortest base-to-base distance first. Since workers are approximately at the same position as their related depot, it would find the optimal worker source.

While the `Architect` handles building the depot like any other structure, the `Settler` module contains expansion logic and orders the construction. It is not an independent module and does not decide when to expand.

Chapter 6

Combat

6.1 Scouting

Scouting is important throughout the entire duration of the game. Early on, players need to know where the opponent is, what faction they are playing and which strategy they are employing. Later on, observing the enemy army size, unit types, base expansions and tech level is important to counter strategies.

Enemy opening -> opponent modeling.

The opponents faction is important when deciding on opening strategies. The factions behave very different early in the game. Openings are a decisively slower when one has to defend against every possible attack. Only if the opponent picked random as a faction will it be hidden. No units are shared across factions, so the first unit discovered will reveal it, which usually is their scout or main-base.

Base positions + expansions

6.1.1 Initial Scouting

The map terrain, along with resource locations, are completely revealed at match start. This includes all possible player start locations. Scouting for the opponent base is then just going through all other start locations. Maps used in competitive matches are usually no larger than four-player size, meaning there are four possible start locations. A player then has to take into account that they might have to scout up to two bases before knowing the enemy start location. Usually a player wants to know more than where the opponent starts, so even when it is deductible where the opponent is, the player wants to scout the base itself.

Enemy opening.

Once the enemy base has been revealed, and with it his opening strategy, the scout is not useful any longer. It is a long trip back to gathering resources, and the scout might be followed, revealing your own location. Harassing enemy workers can put a dent in the enemy economy, even if none of them die. Simply by attacking enemy workers forces the opponent to pull two from gathering to defending. From there, the harassing scout can retreat until it is no longer chased, or lead chasing workers around while attacking passive gatherers. If the scout manages to kill a worker the opponent will fall behind in economy. However, compared to competitive human players, bots are usually too inefficient to take full advantage of this, but every bit helps.

There are additional uses to the scout however compared to harassing. These must all be considered in the grand strategy, as they involve spending resources. These strategies include proxy bunker, photon cannon, barracks or gateway, manner pylons or gas stealing. Proxy structures involve building right below the enemy ramp or even inside their base. The usual distinction here is whether the structures keep the opponent in with defensive towers or rush attack with front line troop producers. Manner pylons are used in conjunction with these as Protoss, where the required pylon for forward bases are placed within the enemy mineral line, blocking and possibly caging enemy workers. Gas stealing involves building a refinery on the enemy geyser, blocking gas harvesting and forcing tier one unit use. While proxy troop production has not seen much use in bots, both gas steal and proxy towers has been used for varying effect.

6.1.1.1 Implementation

Usually build orders include specifically when to send out a scout. In case of Protoss, the scout is often a worker that has just warped in a structure. This would however require a build-order system capable of containing other elements than builds. A simpler solution which is used here is sending a scout when a specific supply limit has been reached. This is noted in build orders, allowing for only slightly inaccurate build-order implementations.

When picking a scout, the bot searches through different worker groups. First the idle, then the mineral miners and finally the gas harvesters. When it comes across a worker not currently carrying any resources, it assigns it as a scout, removing it from the Task Master in the mean time. Contrary to building or defending, scouts are assumed to not return, so it can safely be removed from the local worker pool and harvesting. If the scouting manager is unsuccessful in finding a worker, it retries next frame.

While we have a scout and do not know the opponents position, we pick an unexplored base location and move the scout there. A tile is unexplored if it has never been revealed for the duration of the game. Thus, our home base will not be considered for scouting, and once the scout reaches the destination the tile will be explored, removing it from the possible scouting locations. Exploration is handled by BWAPI.

This implementation scouts as long as no enemy buildings are known, extending its use into late game. Usually fast or cloaked units are used to scout later in the game, but it is not necessary.

6.1.2 Intelligence Management

Locating positions, types.

Geysers <-> refineries

Possible Additions: Faction

6.2 Attacking

When to attack.

Where to attack.

6.2.1 Combat

Predictions.

Grouping.

Prioties.

Retreating.

6.3 Defending

Current: anyone in vicinity

Worker militia.

Anti-scouts / anti-harassment

Worker relocate.

Others: Standing army, Pullback, Vicinity prediction and need.

Chapter 7

Strategy

7.1 Strategies

Main strategies: Rush, Turtle, Boom.

Economic advantage v short term advantage.

Niche strategies: all-in, wall-in, proxies.

7.1.1 Build-orders

Planning.

Openings.

Chapter 8

Conclusion

Agent status, results.

Competitions.

Further study.

Appendix A

Stuff

This appendix is full of stuff ...