# AI Used in Real-Time Strategy Video Game: StarCraft

## Introduction

StarCraft is a real-time strategy video game created by Blizzard in 1997 in which players are given control of buildings and units by which they build an army through an established economy which the surrounding resources constitute. In the game, players build worker units to gather resources, build structures to train attacker units and eventually attack to destroy the opponent. The gameplay involves strategy planning, tactical planning, individual unit control which can be represented as well-defined complex adversarial systems (Certicky, 2013, p. 1).

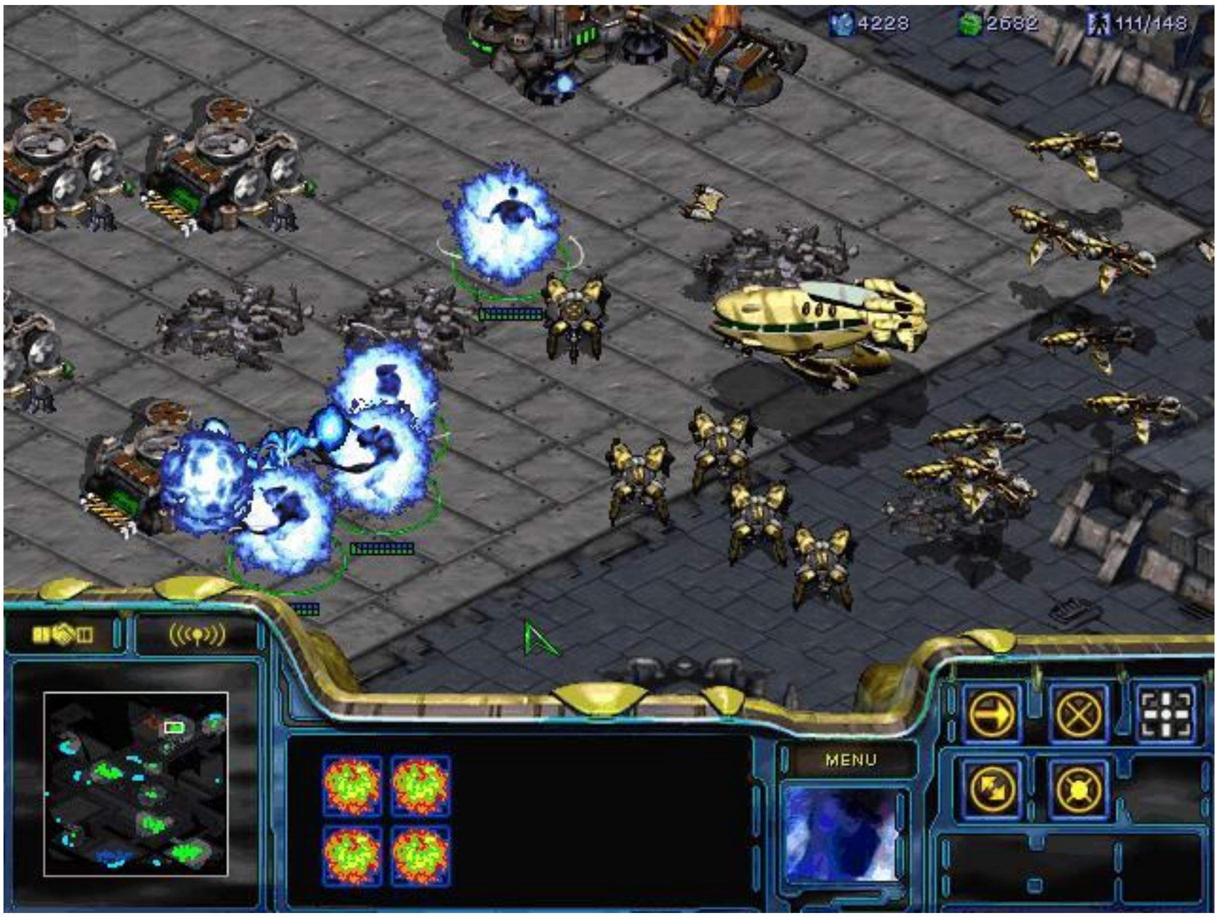


*Figure 1 - A Protoss army (a race in the game) attacking Terran (another race in the game) buildings. Resources gathered (mineral and gas) can be seen in the top right corner.*

## Sub-Problems and Recent Challenges

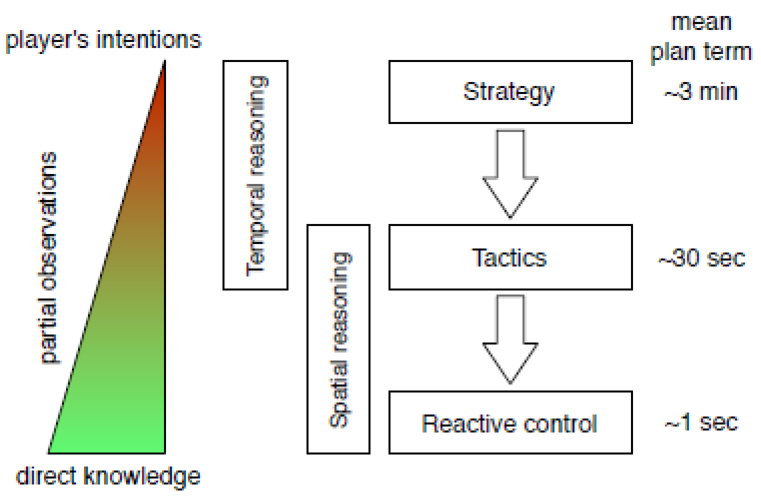
There are lots of sub-problems in which AI techniques are used in this RTS game. The following are the recent challenges in the field (Ontanton et. al, 2013, pp. 3-5):

- **Planning**
  Since the state space is vast, game-tree search is not applicable, therefore multiple level of abstraction is needed: long-term planning for a balance in good economy – large army, short-term planning for combat decisions.

- **Learning**
  Three types of learning: *prior learning, in-game learning, inter-game learning*. Prior learning includes extracting information from replays, or map terrain to gain advantage before the game. In-game learning includes opponent modeling. Inter-game learning concerns about improving the agent from game to game.

- **Uncertainty**
  Map is partially observable, therefore scouting is needed for gathering information about the opponent. Games are adversarial, impossible to predict what opponent will do. Human players consider actions that opponents are likely-to-do.

- **Spatial and temporal reasoning**
  Spatial reasoning: Terrain exploitation: higher ground has advantage over lower ground since units on lower ground has no sight of the higher ground. Deciding where to engage combat to use bottlenecks for advantage. Base expansion, finding and deciding best location according to opponent bases location.
  Temporal reasoning: timing attacks, retreats. Long-term planning of actions that has a higher impact on economy such as upgrades, strategy switching etc.



- **Domain knowledge exploitation**
  Traditional strategy games (chess, go) exploited large amounts of the domain knowledge and developed good evaluation functions. On the other hand StarCraft has a much larger domain and approaches are in two ways: Hard-coding the strategies into agents and running an algorithm to decide instead of adaptive approach; large set of replays are created and agents are trying to learn from replays. Both are still open problems.

- **Task decomposition**
  Strategy: high level decision making: finding an efficient or counter strategy.
  Tactics: Implementation of the current strategy: army composition, building positioning, army movements, timing etc. Tactics concern a group of units.
  Reactive Control: Micro-management: Firing, retreating, kiting (hit-and-run).
  Terrain Analysis: Choke-points, mineral and gas (resource) locations, high grounds.
  Intelligence Gathering: Scouting the opponent.

For the human side of decision making, the StarCraft community mentions two tasks:

- Micro management: micro-management roughly corresponds to the reactive control which involves the individual or a small group of unit control and positioning and etc.
- Macro management: macro covers all of the above except reactive control. Unit producing, expanding the base for extra resource income at the right time, choosing the appropriate unit combination to counter the opponent etc.

Current studies in RTS game AI divided these tasks into three parts as shown in the figure above: strategy, tactics and reactive control.

## Sub-Problem: Fast Heuristic Search Used In Combat Scenarios

Having mentioned the domain of the field, this paper will be focusing on game combat scenarios. This sub-problem focuses on battle unit management (mentioned as micromanagement or reactive control above). Micromanagement is a crucial skill for a player to optimize combat outcomes which gives an opportunity to yield victory on a combat in which two players have same size of an army.

Although the game itself is not captured, the following video visualizes by a simulation software how micromanagement works in some combat scenarios. Since this is not an actual game footage, the reader would not associate what's happening in the video with the micromanagement term. If that is the case, watch the other videos for a better idea. This is the simulation video link: http://www.youtube.com/watch?v=ixkJhCLtesQ

This video is from the StarCraft II (all the previous footage is from the original StarCraft (1997). StarCraft II was released in 2010. Albeit the differences, the micromanagement idea is the same, only mechanics change a little) captured during World Championship Series Season 2 and players are human: http://youtu.be/4qr0a9QfTiM?t=26s 0:26 – 1:04

If the previous video did not ring any bell to the reader, this video of StarCraft II in which some combat scenarios are simulated by the game engine (which results in very accurate game scenarios) will show reader the difference between a micromanaged-combat and an all-units-on-attack-command combat. http://youtu.be/YZM4u99GDTE?t=1m37s 1:37 – as much as the reader wants to watch.

To sum up, micromanagement is the controlling of individual or very small group of units during a combat to achieve better results in a combat scenario as opposed to just giving all units the attack command where every unit will attack the closest aggressive opponent unit.

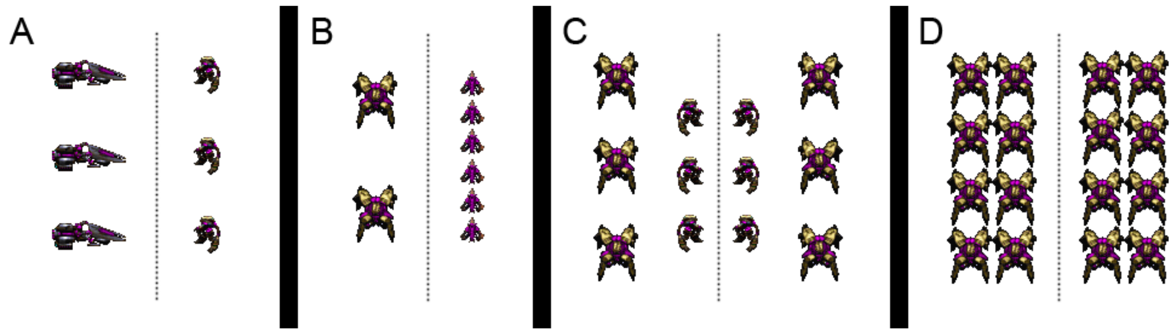


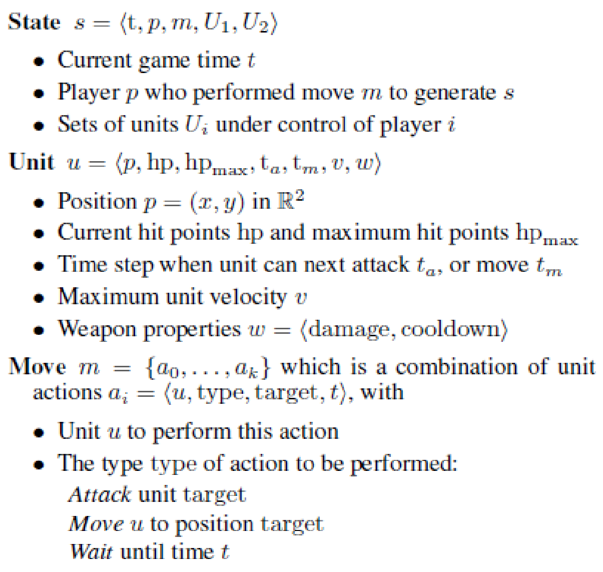
*Figure 2- Four combat scenarios where armies have approximate firepower. The better micromanagement will yield victory.*

The battles between armies in RTS games fit well into classification of two-player zero-sum simultaneous move games (Churchill & Buro, 2012, p. 4). Therefore simple scripted strategy over battles will not yield the best performance always. Churchill and Buro implemented the ABCD (Alpha-Beta search Considering Durations) algorithm and integrated it into their game-playing agent: UAlbertaBot. They explain the algorithm as follows:

*“ABCD search takes action durations into account and approximates game-theoretic solutions of simultaneous move games by sequentializing moves using simple policies such as Max-Min-Max-Min or Min-Max-Min-Max.”*

## The Combat Model and Solution Concepts

There are three main components are designed for the modeling: states, units and moves (Churchill, Saffidine & Buro, 2012, p. 2).



**State** $s = \langle \mathrm{t}, p, m, U_1, U_2 \rangle$

- Current game time $t$
- Player $p$ who performed move $m$ to generate $s$
- Sets of units $U_i$ under control of player $i$

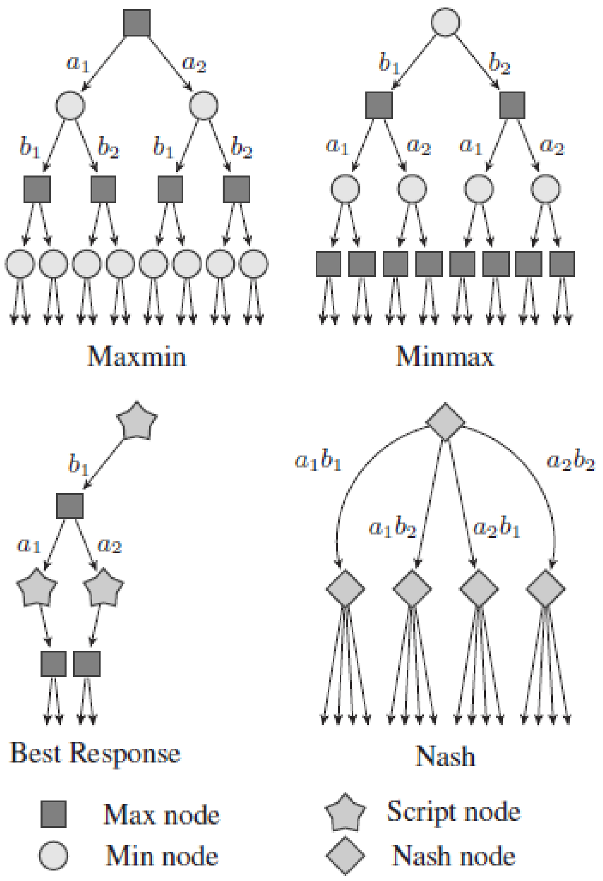
**Unit** $u = \langle p, \mathrm{hp}, \mathrm{hp}_{\max}, \mathrm{t}_a, \mathrm{t}_m, v, w \rangle$

- Position $p = (x, y)$ in $\mathbb{R}^2$
- Current hit points hp and maximum hit points $\mathrm{hp}_{\max}$
- Time step when unit can next attack $t_a$, or move $t_m$
- Maximum unit velocity $v$
- Weapon properties $w = \langle \mathrm{damage}, \mathrm{cooldown} \rangle$

**Move** $m = \{a_0, \ldots, a_k\}$ which is a combination of unit actions $a_i = \langle u, \mathrm{type}, \mathrm{target}, t \rangle$, with

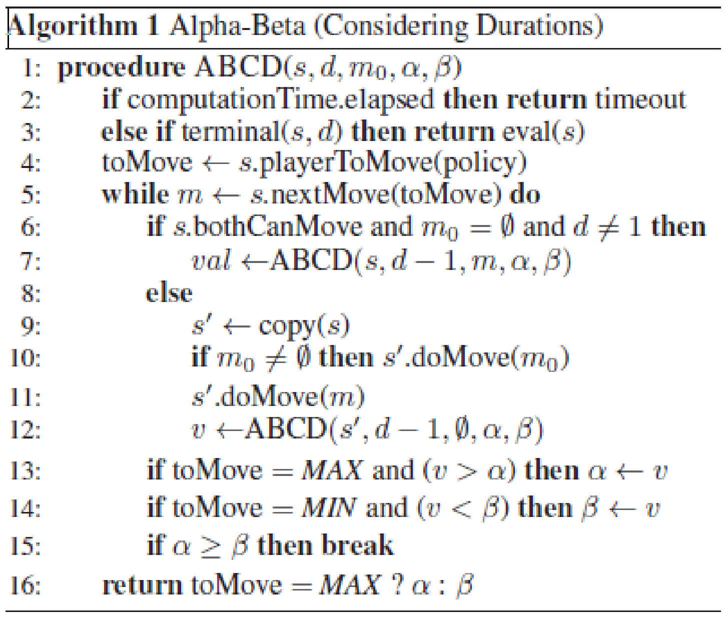
- Unit $u$ to perform this action
- The type type of action to be performed:
  *Attack* unit target
  *Move* $u$ to position target
  *Wait* until time $t$

Given this model, the termination criterion and utility functions can be formed accordingly. A terminal position is the position in which a player’s all units have reached 0 hp or if a certain amount of time has passed (measured in game-frames or unit actions). Combining the utility functions, termination criterion and combat model results in *combat games* which are classified as zero-sum games. This property together with the fully observable state variables and simultaneous moves places combat games in the class of “stacked matrix games” (Churchill, Saffidine & Buro, 2012, p. 2). Those games can be solved by backward induction starting with terminal states via Nash equilibrium computations.



Since the state-space is vast in RTS games, decisions have to be made quickly (during a single simulation frame, i.e. 50 ms). This makes computing optimal moves impossible for middle-large scale combats, which are generally the case in the game combat scenarios.

The ABCD algorithm's approach is to declare nodes to be leaf nodes once a certain depth limit is reached. In leaf nodes MAX's utility is then estimated by calling an evaluation function and this value is propagated up the tree like true terminal node utilities (Churchill, Saffidine & Buro, 2012, p. 3).



**Algorithm 1** Alpha-Beta (Considering Durations)

```
procedure ABCD($s, d, m_0, \alpha, \beta$)
    if computationTime.elapsed then return timeout
    else if terminal($s, d$) then return eval($s$)
    toMove ← $s$.playerToMove(policy)
    while $m \leftarrow s$.nextMove(toMove) do
        if $s$.bothCanMove and $m_0 = \emptyset$ and $d \neq 1$ then
            $val \leftarrow$ABCD($s, d-1, m, \alpha, \beta$)
        else
            $s' \leftarrow$ copy($s$)
            if $m_0 \neq \emptyset$ then $s'$.doMove($m_0$)
            $s'$.doMove($m$)
            $v \leftarrow$ABCD($s', d-1, \emptyset, \alpha, \beta$)
        if toMove = $MAX$ and ($v > \alpha$) then $\alpha \leftarrow v$
        if toMove = $MIN$ and ($v < \beta$) then $\beta \leftarrow v$
        if $\alpha \geq \beta$ then break
    return toMove = $MAX$ ? $\alpha$ : $\beta$
```

## References

- Certicky, M., (2013). *Implementing a Wall-In Building Placement in StarCraft with Declarative Programming*, retrieved January, 2, from http://arxiv.org/abs/1306.4460
- Ontanon S., Synnaeve G., Uriarte A., Richoux F., Churchill D. and Preuss M., (2013)*. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft*. Computational Intelligence and AI in Games, IEEE Transactions on (Volume: 5, Issue: 4)
- Churchill D., Buro M., (2012). *Incorporating Search Algorithms into RTS Game Agents.* AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment; Eighth Artificial Intelligence and Interactive Digital Entertainment Conference
- Churchill D., Saffidine A. and Buro M., (2012). *Fast Heuristic Search for RTS Game Combat Scenarios*. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment Eighth Artificial Intelligence and Interactive Digital Entertainment Conference