A Multi-Agent Potential Field-Based Bot for a Full RTS Game Scenario

Johan Hagelbäck and Stefan J. Johansson
School of Computing
Blekinge Institute of Technology
Box 520, SE-372 25, Ronneby, Sweden
email: sja@bth.se, jhg@bth.se

Abstract

Computer games in general, and Real Time Strategy games in particular is a challenging task for both AI research and game AI programmers. The player, or AI bot, must use its workers to gather resources. They must be spent wisely on structures such as barracks or factories, mobile units such as soldiers, workers and tanks. The constructed units can be used to explore the game world, hunt down the enemy forces and destroy the opponent buildings. We propose a multi-agent architecture based on artificial potential fields for a full real time strategy scenario. We validate the solution by participating in a yearly open real time strategy game tournament and show that the bot, even though not using any form of path planning for navigation, is able to perform well and win the tournament.

Keywords

Agents:Swarm Intelligence and Emergent Behavior, Multidisciplinary Topics and Applications:Computer Games

Introduction

There are many challenges for a real-time strategy (RTS) bot. The bot has to control a number of units performing tasks such as gathering resources, exploring the game world, hunting down the enemy and defend own bases. In modern RTS games, the number of units can in some cases be up to several hundred. The highly dynamic properties of the game world (e.g. due to the large number of moving objects) make navigation sometimes difficult using conventional pathfinding methods. Artificial Potential Fields, an area originating from robotics, has been used with some success in video games. Thurau et al. has developed a game bot which learns behaviours in the First-Person Shooter game Quake II through imitation (Thurau, Bauckhage, and Sagerer 2004). The behavious are represented as attractive potential fields placed at interesting points in the game world, for example choke points or areas providing cover. The strength of the fields are increased/decreased by observing a human player.

Multi-agent Potential Fields

In previous work we proposed a methodology for designing a multi-agent potential fields (MAPF) based bot in a real-time strategy game environment (Hagelbäck and Johansson 2008b). The methodology involved the following six steps: *i) Identifying the objects, ii) Identifying the fields, iii) Assigning the charges, iv) Deciding on the granularities, v) Agentifying the core objects* and *vi) Construct the* MAS *architecture.* For further details on the methodology, we refer to the original description (Hagelbäck and Johansson 2008b). In this paper we use the methodology to build a bot for the full RTS game scenario.

ORTS

Open Real Time Strategy (ORTS) (Buro 2007) is a real-time strategy game engine developed as a tool for researchers within AI in general and game AI in particular. ORTS uses a client-server architecture with a game server and players connected as clients. Users can define different types of games in scripts where units, structures and their interactions are described. All types of games from resource gathering to full real time strategy (RTS) games are supported.

In previous work we used the proposed methodology to develop a MAPF based bot for the quite simple game type *Tankbattle* (Hagelbäck and Johansson 2008b; 2008a). Here, we extend the work to handle the more complex Full RTS game (Buro 2007). In this game, two players start with five workers and a control center each. The *workers* can be used to gather resources from nearby mineral patches, or to construct new control centers, barracks or factories. A *control center* serves as the drop point for resources gathered by workers, and it can produce new workers as well. *Barracks* are used to construct *marines;* light-weight combat units. If a player has at least one barrack, it can construct a *factory*. Factories are used to construct *tanks;* heavy combat units with long firerange. A player wins by destroying all the buildings of the opponent. The game also contains a number of neutral units called *sheep.* These are small indestructible units moving randomly around the map making pathfinding and collision detection more complex. Both games are part of the annual ORTS tournament organised by the University of Alberta (Buro 2007).

MAPF in a Full RTS Scenario

We have implemented a MAPF based bot for playing the Full RTS game in ORTS following the proposed steps. Since this

work extends previous research on MAPF based bots (and the space limitations prevents us from describing everything in detail), we will concentrate this on the additions we have made. For the details about the MAPF methodology and the Tankbattle scenario, we refer to (Hagelbäck and Johansson 2008b; 2008a).

Identifying objects

We identify the following objects in our application: *Workers, Marines, Tanks, Control centers, Barracks, Factories, Cliffs,* and the neutral *Sheep,* and *Minerals.* Units and buildings are present on both sides.

Identifying fields

In the Tankbattle scenario we identified four tasks: Avoid colliding with moving objects, Hunt down the enemy's forces, Avoid colliding with cliffs, and Defend the bases (Hagelbäck and Johansson 2008b). In the Full RTS scenario we identify the following additional tasks: Mine resources, Create buildings, Train workers and marines, Construct tanks, and Explore the game world. The tasks are organised into the following types of potential fields:

Field of Navigation. This field contains all objects that have an impact on the navigation in the game world: *terrain, own units and buildings, minerals* and *sheep.* The fields are repelling to avoid that our agents collide with the obstacles.

Strategic Field. This field contains the goals for our agents and is an attractive field, different for each agent type. Tanks have attractive fields generated by opponent units and buildings. Workers mining resources have attractive fields generated by mineral patches (or if they cannot carry anymore, the control center where it can drop them off).

Field of Exploration. This field is used by workers assigned to explore the game world and attract them to unexplored areas.

Tactical field. The purpose of the tactical field is to coordinate movements between our agents. This is done by placing a temporary small repelling field at the next movement position for an agent. This prevents own units from moving to the same location if there are other routes available.

Field of spatial planning. This field helps us finding suitable places on the map to construct new buildings such as control centers, barracks and factories at.

This approach has similarities with the work by Paul Tozour in (Tozour 2004), where the author describes multiple layers of influence maps. Each layer is responsible for handling one task, for example the distance to static objects or the line-of-fire of own agents. The different fields sum up to form a total field that is used as a guide for the agents when selecting actions.

Assigning charges and granularity

Each game object that has an effect on navigation or tactics for our agents has a set of charges which generate a potential field around the center of the object. All fields generated by objects are weighted and summed to form a total field which is used by agents when selecting actions. The initial set of charges was hand crafted. However, the order of importance between the objects simplifies the process of finding good values and the method seems robust enough to allow the bot to work good anyhow. Below is a detailed description of each field. As in the Tankbattle scenario described in (Hagelbäck and Johansson 2008a), we use a granularity of 1x1 game world points for potential fields, and all dynamic fields are updated every frame.

The opponent units. Opponent units, tanks marines and workers, generate different fields depending on the agent type and its internal state. In the case of own attacking units, tanks and marines, the opponent units generate attracting symmetric surrounding fields where the highest potentials are at radius equal to the maximum shooting distance, MSD from the enemy unit. This is illustrated in Figure 1. It shows a tank (black circle) moving to attack an opponent unit E. The highest potentials (light grey areas) are located in a circle around E.

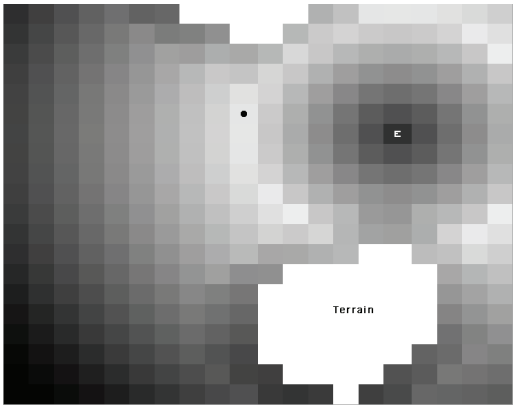

Figure 1: A tank (black circle) engaging an opponent unit E. Light grey areas have higher potential than darker grey areas.

After an attacking unit has fired its weapon the unit enters a cooldown period when it cannot attack. This cooldown period may be used to retreat from enemy fire, which has shown to be a successful strategy (Hagelbäck and Johansson 2008a). In this case the opponent units generate repelling fields with radius slightly larger than the MSD. The use of a defensive field makes our agents surround the opponent unit cluster at MSD even if the opponent units pushes our agents backwards. This is illustrated in Figure 2. The opponent unit E is now surrounded by a strong repelling field that makes the tank (white circle) retreat outside MSD of the opponent.

The fields generated by game objects are different for different types of own units. In Figure 1 a tank is approaching an enemy unit. A tank typically has longer fire range than for example a marine. If a marine would approach the enemy unit a field where the highest potentials are closer to the enemy unit would be generated. Below is pseudo-code for calculating the potential an enemy object e generates in a point p in the game world.

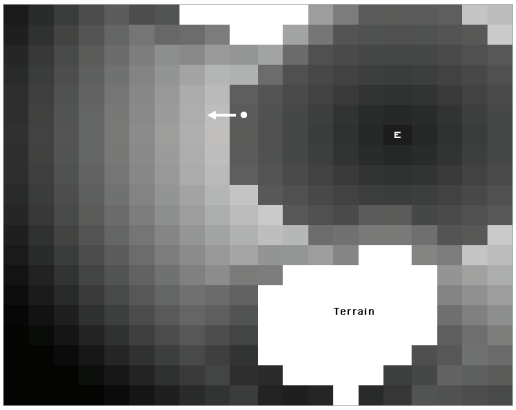

Figure 2: A tank (white circle) in cooldown retreats outside the MSD of an opponent unit.

$distance =$ distanceBetween(Position p, EnemyObject e);
$potential =$ calculatePotential($distance$,
OwnObjectType ot, EnemyObjectType et);

Own buildings. Own buildings, control centers barracks and factories, generate repelling fields for obstacle avoidance. An exception is in the case of workers returning minerals to a control center. In this case control centers generate an attractive field calculated using Equation 2. The repelling potential $p_{ownB}(d)$ at distance d from the center of the building is calculated using Equation 1.

$$p_{ownB}(d) = \begin{cases} 6 \cdot d - 258 & \text{if } d <= 43 \\ 0 & \text{if } d > 43 \end{cases} \quad (1)$$

$$p_{attractive}(d) = \begin{cases} 240 - d \cdot 0.32 & \text{if } d <= 750 \\ 0 & \text{if } d > 750 \end{cases} \quad (2)$$

Minerals. Minerals generate two different types of field; one attractive field used by workers mining resources and a repelling field that is used for obstacle avoidance. The potential $p_{attractive}(d)$ at distance d from the center of a mineral is calculated using Equation 2. In the case when minerals generate a repelling field, the potential $p_{mineral}(d)$ at distance d from the mineral is calculated as:

$$p_{mineral}(d) = \begin{cases} -20 & \text{if } d <= 8 \\ 20 - 2 \cdot d & \text{if } d \in]8, 10] \end{cases} \quad (3)$$

Figure 3 and 4 illustrates a worker mining resources from a nearby mine. In Figure 3 the worker is ordered to gather more resources and an attractive potential field is placed around the mine. Terrain, own worker units and the base all generate small repelling fields used for obstacle avoidance. When the worker has gathered as much resources it can carry, it must return to the base to drop them off. This is shown in Figure 4. The attractive charge is now placed in the center of the base, and the mine now generates a small repelling field for obstacle avoidance.

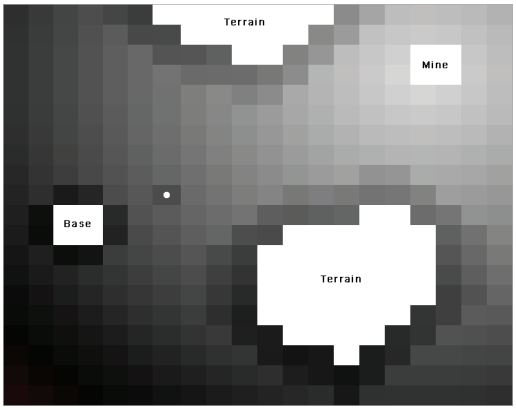

Figure 3: A worker unit (white circle) moving towards a mine to gather resources. The mine generates an attractive field and mountains (black) generate small repelling fields for obstacle avoidance. Light grey areas are more attracting than darker grey areas.

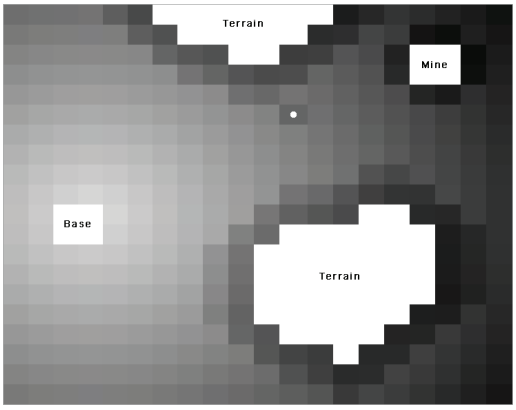

Figure 4: A worker unit (white circle) moving towards a base to drop of gathered resources.

Field of exploration. The field of exploration is a field with attractive charges at the positions in the game world that need to be explored. First an importance value for each terrain tile is calculated in order to find next position to explore. This process is described below. Once a position is found, the Field of Navigation, Equation 4, is used to guide the unit to the spot. This approach seems to be more robust than letting all unexplored areas generate attractive potentials. In the latter case explorer units tend to get stuck somewhere in the middle of the map due to the attractive potentials generated from unexplored areas in several directions.

$$p_{navigation}(d) = \begin{cases} 150 - d * 0.1 & \text{if } d <= 1500 \\ 0 & \text{if } d > 1500 \end{cases} \quad (4)$$

The importance value for each tile is calculated as follows:

1. Each terrain tile (16x16 points) is assigned an explore value, $E(x, y)$, initially set to 0.

2. In each frame, $E(x, y)$ is increased by 1 for all passable tiles.
3. If a tile is visible by one or more of our own units in the current frame, its $E(x, y)$ is reset to 0.
4. Calculate an importance value for each tile using Equation 5. The distance d is the distance from the explorer unit to the tile.

$$importance(x, y, d) = 2.4 \cdot E(x, y) - 0.1d \quad (5)$$

Figure 5 illustrates a map with a base and an own explorer unit. The white areas of the map are unexplored, and the areas visible by own units or buildings are black. The grey areas are previously explored areas that currently are not visible by own units or buildings. Light grey tiles have higher explore values than darker grey tiles.

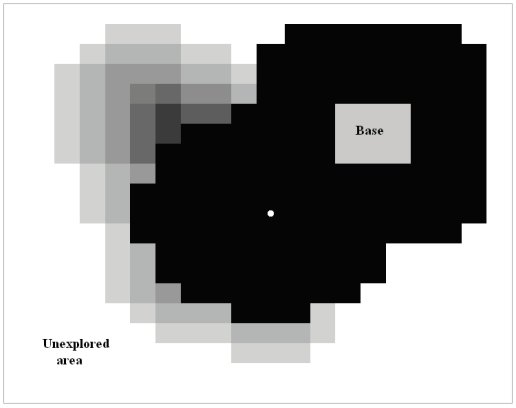

Figure 5: Explore values as seen by the explorer unit (white circle). Grey areas have previously been visited. Black areas are currently visible by an own unit or building.

The next step is to pick the tile of the greatest importance (if there are several equally important, pick one of them randomly), and let it generate the field. This is shown in Figure 6. The explorer unit move towards the choosen tile from Figure 5 to explore next.

Base building. When a worker is assigned to construct a new building, a suitable build location must first be found. The method used to find the location is described in the SpatialPlanner agent section below. Once a location is found, the potential $p_{builder}(d)$ at distance d from the position to build at is calculated using the Field of Navigation (see Equation 4).

The agents of the bot

Each own unit (worker, marine or tank) is represented by an agent in the system. The multi-agent system also contains a number of agents not directly associated with a physical object in the game world. The purpose of these agents is to coordinate own units to work towards common goals (when applicable) rather than letting them act independently. Below follows a more detailed description of each agent.

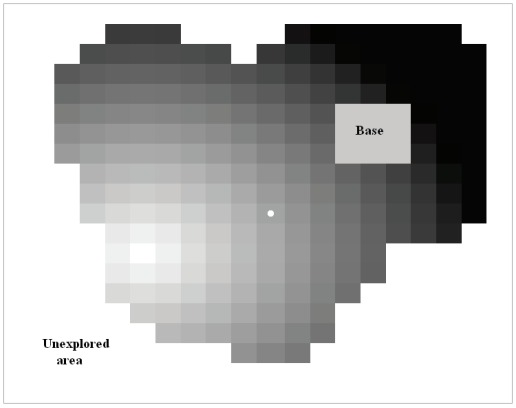

Figure 6: The explorer unit (white circle) move towards the tile with the highest importance value (light grey area).

CommanderInChief. The CommanderInChief agent is responsible for making an overall plan for the game, called a battleplan. The battleplan contains the order of creating units and buildings, for example start with training 5 workers then build a barrack. It also contains special actions, for example sending units to explore the game world. When one post in the battleplan is completed, the next one is executed. If a previously completed post no longer is satisfied, for example a worker is killed or a barrack is destroyed, the CommanderInChief agent takes the necessary actions for completing that post before resuming current actions. For a new post to be executed there must be enough resources available. The battleplan is based on the ideas of subsumption architectures (see (Brooks 1986)) shown in Figure 7. Note that all workers, unless ordered to do something else, are gathering resources.

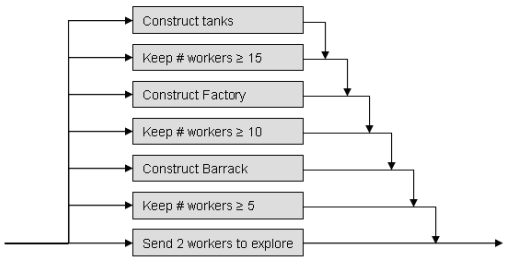

Figure 7: The subsumption hierarchy battleplan.

CommanderInField. The CommanderInField agent is responsible for executing the battleplan generated by the CommanderInChief. It sets the goals for each unit agent, and change goals during the game if necessary (for example use a worker agent currently gathering resources to construct a new building, and to have the worker go back to resource gathering after the building is finished). The CommanderInField agent has three additional agents to help it with the execution of the battleplan; GlobalMapAgent, AttackCoordinator and SpatialPlanner.

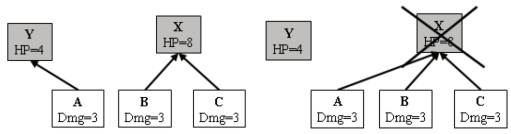

Figure 8: Attacking the most damaged unit (to the left) vs. Optimize attacks (to the right).

GlobalMapAgent. In ORTS a data structure with the current game world state is sent each frame from the server to the connected clients. The location of buildings are however only included in the data structure if an own unit is within visibility range of the building. It means that an enemy base that has been spotted by an own unit and that unit is destroyed, the location of the base is no longer sent in the data structure. Therefore our bot has a dedicated global map agent to which all detected objects are reported. This agent always remembers the location of previously spotted enemy bases until a base is destroyed, as well as distributes the positions of detected enemy units to all the own units.

AttackCoordinator. The purpose of the attack coordinator agent is to optimize attacks at enemy units. The difference between using the coordinator agent compared to attacking the most damaged unit within fire range (which seemed to be the most common approach used in the 2007 years' ORTS tournament) is best illustrated with an example. A more detailed description of the attack coordinator can be found in (Hagelbäck and Johansson 2008b).

In Figure 8 the own units A, B and C deals 3 damage each. They can all attack opponent unit X (X can take 8 more damage before it is destroyed) and unit A can also attack unit Y (Y can take 4 more damage before it is destroyed). If an *attack the weakest* strategy is used, unit A will attack Y, and B and C will attack X with the result that both X and Y will survive. By letting the coordinator agent optimize the attacks, all units are coordinated to attack X, which then is destroyed and only Y will survive.

SpatialPlanner. To find a suitable location to construct new buildings at, we use a special type of field only used to find a spot to build at. Once it has been found by the Spatial Planning Agent, a worker agent uses the Field of Navigation (see Equation 4) to move to that spot. Below follow equations used to calculate the potential game objects generate in the spatial planning field. *Own buildings*. Own buildings generate a field with an inner repelling area (to avoid construct buildings too close to each other) and an outer attractive area (for buildings to be grouped together). Even though the size differs somewhat between buildings for simplicity we use the same formula regardless of the type of building. The $p_{ownbuildings}(d)$ at distance d from the center of an own building is calculated as:

$$p_{ownbuildings}(d) = \begin{cases} -1000 & \text{if } d <= 115 \\ 230 - d & \text{if } d \in]115, 230] \\ 0 & \text{if } d > 230 \end{cases} \quad (6)$$

Enemy buildings. Enemy buildings generate a repelling field. The reason is of course that we do not want own buildings to be located too close to the enemy. The $p_{enemybuildings}(d)$ at distance d from the center of an enemy building is calculated as:

$$p_{enemybuildings}(d) = \begin{cases} -1000 & \text{if } d <= 150 \\ 0 & \text{if } d > 150 \end{cases} \quad (7)$$

Minerals. It is not possible to construct buildings on top of minerals therefore they have to generate repelling fields. The $p_{mineral}(d)$ at distance d from the center of a mineral is calculated using Equation 8. The field is slightly attractive outside the repelling area since it is beneficial to have bases located close to resources.

$$p_{mineral}(d) = \begin{cases} -1000 & \text{if } d <= 90 \\ 5 - d \cdot 0.02 & \text{if } d \in]90, 250] \\ 0 & \text{if } d > 250 \end{cases} \quad (8)$$

Impassable terrain. Cliffs generate a repelling field to avoid workers trying to construct a building too close to a cliff. The $p_{cliff}(d)$ at distance d from the closest cliff is calculated as:

$$p_{cliff}(d) = \begin{cases} -1000 & \text{if } d <= 125 \\ 0 & \text{if } d > 125 \end{cases} \quad (9)$$

Game world edges. The edges of the game world have to be repelling as well to avoid workers trying to construct a building outside the map. The $p_{edge}(d)$ at distance d from the closest edge is calculated as:

$$p_{edge}(d) = \begin{cases} -1000 & \text{if } d < 90 \\ 0 & \text{if } d >= 90 \end{cases} \quad (10)$$

To find a suitable location to construct a building at, we start by calculating the total buildspot potential in the current position of the assigned worker unit. In the next iteration we calculate the buildspot potential in points at a distance of 4 tiles from the location of the worker, in next step at distance 8, and continue up to distance 200. The position with the highest buildspot potential is the location to construct the building at. Figure 9 illustrates the field used by the Spatial Planner Agent to find a spot for the worker (black circle) to construct a new building at. Lighter grey areas are more attractive than darker grey areas. The location to construct the building at is shown as a black non-filled rectangle. Once the spot is found the worker agent uses the Field of Navigation to move to that location.

Experiments

We used the ORTS tournament of 2008 as a benchmark to test the strength of our bot. The number of participants in the Full RTS game was unfortunately very low, but the results are interesting anyway since the opponent team from University of Alberta has been very competitive in earlier tournaments. The UOFA bot uses a hierarchy of commanders where each major task such as gathering resources or

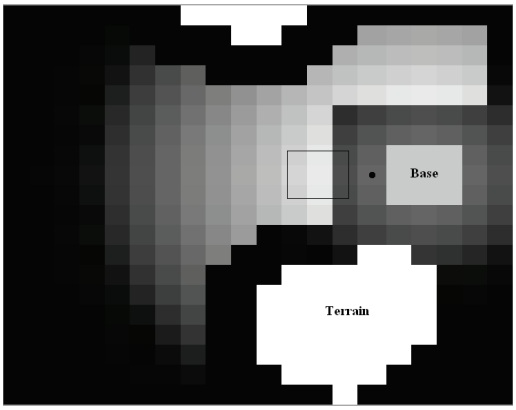

Figure 9: Field used by he Spatial Planner agent to find a build spot (black non-filled rectangle).

building a base is controlled by a dedicated commander. The Attack commander, responsible for hunting down and destroy enemy forces, gather units in squads and uses A* for the pathfinding. The results from the tournament are shown in Table 1. Our bot won 82.5% of the games against the opponent team over 2x200 games (200 different maps where the players switched sides).

Team	Win %	Wins/games	DC
Blekinge	82.5%	(330/400)	0
Uofa	17.5%	(70/400)	3

Table 1: Results from the ORTS tournament of 2008. DC is the number of disconnections due to client software failures.

Discussion

There are several interesting aspects here.

- First, we show that the approach we have taken, to use Multi-agent potential fields, is a viable way to construct highly competitive bots for RTS scenarios of medium complexity. Even though the number of competitors this year was very low, the opponent was the winner (89 % wins) of the 2007 tournament. Unfortunately, ORTS server updates have prevented us from testing our bot against the other participant of that year, but there are reasons to believe that it would manage well against those solutions too (although it is not sure, since the winning relation between strategies in games is not transitive, see e.g. Rock, Paper Scissors (deJong 2004)). We argue though that the use of open tournaments as a benchmark is still better than if we constructed the opponent bots ourselves.
- Second, we combine the ideas of using a role-oriented MAS architecture and MAPF bots.
- Third, we introduce (using the potential field paradigm) a way to place new buildings in RTS games.

Conclusions and Future Work

We have constructed an ORTS bot based on both the principles of role-oriented MAS and Multi-agent Potential Fields. The bot is able to play an RTS game and outperforms the competitor by winning more than 80% of the games in an open tournament where it participated.

Future work will include to generate a battleplan for each game depending on the skill and the type of the opponent it is facing. The strategy of our bot is now fixed to construct as many tanks as possible to win by brute strength. It can quite easily be defeated by attacking our base with a small force of marines before we are able to produce enough tanks. The CommanderInChief agent should also be able to change battleplan to adapt to changes in the game to, for example, try to recover from an attack by a marine force early in the game. Our bot is also set to always directly rebuild a destroyed building. If, for example, an own factory is destroyed it might not be the best option to directly construct a new one. It might be better to train marines and/or move attacking units back to the base to get rid of the enemy units before constructing a new factory. There are also several other interesting techniques to replace the sum-sumption architecture.

We believe that a number of details in the higher level commander agents may improve in the future versions when we better adapt to the opponents. We do however need more opponent bots that uses different strategies to improve the validation.

References

Brooks, R. 1986. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, RA–2(1):14–23.

Buro, M. 2007. ORTS — A Free Software RTS Game Engine. http://www.cs.ualberta.ca/∼mburo/orts/ URL last visited on 2009-03-26.

deJong, E. 2004. Intransitivity in coevolution. In Parallel Problem Solving from Nature - PPSN VIII. In *volume 3242 of Lecture Notes in Computer Science*. Springer.

Hagelbäck, J., and Johansson, S. J. 2008a. The Rise of Potential Fields in Real Time Strategy Bots. In *4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Hagelbäck, J., and Johansson, S. J. 2008b. Using Multi-agent Potential Fields in Real-Time Strategy Games. In *L. Padgham and D. Parkes editors, Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.

Thurau, C.; Bauckhage, C.; and Sagerer, G. 2004. Learning human-like movement behavior for computer games. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)*.

Tozour, P. 2004. Using a Spatial Database for Runtime Spatial Analysis. In *AI Game Programming Wisdom 2*. Charles River Media.