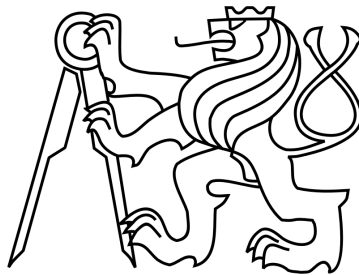


CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING



## Master's Thesis

Methods of multi-agent movement control and  
coordination of groups of mobile units in a  
real-time strategy games

**Author:** Serhij Lebedynskyj

**Thesis supervisor:** Šusta Richard Ing., Ph.D.

Prague, 2/2/15

**Název práce:** Metody Multiagentní řízení pohybu a koordinace skupin mobilních jednotek ve strategických hrách reálného času

**Autor:** Serhij Lebedynskij

**Katedra (ústav):** Katedra řídicí techniky

**Vedoucí bakalářské práce:** Šusta Richard Ing., Ph.D.

**e-mail vedoucího:** [susta@fel.cvut.cz](mailto:susta@fel.cvut.cz)

**Abstrakt** Tato práce nabízí metodu pro reaktivní řízení jednotek v real-time strategické (RTS) počítačové hře pomocí multi-agentních potenciálových polí. Klasická RTS hra StarCraft: Broodwar byla vybrána jako testovací platforma díky jejímu postavení na konkurenční scéně umělé inteligence (UI). Nabízená umělá inteligence ovládá své jednotky pomocí umístění různých potenciálových polí na objekty a na místa v herním světě. Snahou této práce je vylepšit předchozí metody využívající potenciálová pole.

---

**Title:** Methods of multi-agent movement control and coordination of groups of mobile units in a real-time strategy games

**Author:** Serhij Lebedynskij

**Department:** Department of control engineering

**Supervisor:** Šusta Richard Ing., Ph.D.

**Supervisor's e-mail address:** [susta@fel.cvut.cz](mailto:susta@fel.cvut.cz)

**Abstract** This thesis proposes an approach to Reactive Control in Real-Time Strategy (RTS) computer games using Multi-Agent Potential Fields. The classic RTS title StarCraft: Broodwar has been chosen as testing platform due to its status in the competitive Artificial Intelligence (AI) scene. The proposed AI controls its units by placing different types of potential fields in objects and places around the game world. This work is an attempt to improve previous methods done with Potential Field in RTS.

České vysoké učení technické v Praze  
Fakulta elektrotechnická

katedra řídicí techniky

## ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Serhij Lebedynskij**

Studijní program: Kybernetika a robotika  
Obor: Systémy a řízení

Název tématu: **Metody Multiagentní řízení pohybu a koordinace skupin mobilních jednotek ve strategických hrách reálného času**

Pokyny pro vypracování:

1. Seznamte se s problematikou řízení jednotek ve strategické hře reálného času StarCraft.
2. Vypracujte řešerši používaných přístupů pro řízení pohybu jednotek pro efektivní splnění mise.
3. Navrhněte řešení problému řízení jednotek založené na odhadu vhodných akcí z dostupných informací o prostředí a aktivity jednotek protivníka.
4. Navržené řešení implementuje a porovnejte s vybranými přístupy řešení.

Seznam odborné literatury:

- [1] Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A survey of real-time strategy game ai research and competition in starcraft.
- [2] Synnaeve, G., & Bessiere, P. (2011, August). A Bayesian model for RTS units control applied to StarCraft. In \*Computational intelligence and Games (CIG), 2011 IEEE Conference on\* (pp. 190-196). IEEE.
- [3] Parra, R., & Garrido, L. (2013). Bayesian networks for micromanagement decision imitation in the RTS game starcraft. In \*Advances in Computational Intelligence\* (pp. 433-443). Springer Berlin Heidelberg.
- [4] Rathe, E. A., & Svendsen, J. B. (2012). Micromanagement in Starcraft using potential fields tuned with a multi-objective genetic algorithm.

Vedoucí: Ing. Richard Šusta, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016



V Praze dne 25. 2. 2015

# Contents

<b>1. Introduction</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Problem Statement . . . . .	3
1.3. Thesis Structure . . . . .	3
<b>2. Background and Related Work</b>	<b>4</b>
2.1. Real-time strategy games . . . . .	4
2.2. StarCraft . . . . .	5
2.3. Challenges in RTS Game AI . . . . .	5
2.4. Existing work on RTS Game AI . . . . .	6
2.4.1. Tactics . . . . .	6
2.4.2. Reactive control . . . . .	7
<b>3. Platform Setup</b>	<b>11</b>
3.1. StarCraft . . . . .	11
3.2. BWAPI . . . . .	11
3.2.1. Physical Properties . . . . .	12
3.2.2. Unit control methods . . . . .	12
3.3. System Requirements . . . . .	13
<b>4. Methodology</b>	<b>14</b>
4.1. Potential Fields . . . . .	14
4.2. Multi-Agent Potential Fields . . . . .	17
4.3. Bayesian function . . . . .	17
4.4. Finite State Machine . . . . .	18
4.5. Fuzzy logic . . . . .	19
4.5.1. Fuzzyfication . . . . .	19
4.5.2. Evaluation of fuzzy rules . . . . .	20
<b>5. Agent Implementation</b>	<b>21</b>
5.1. Multi-Agent Potential Fields . . . . .	21
5.1.1. Identifying object . . . . .	21
5.1.2. Identifying fields . . . . .	21
5.1.3. Assigning charges to the objects . . . . .	22
5.1.4. Assigning granularity of time and space in the environment . . . . .	33
5.1.5. The agent of the system . . . . .	34

5.1.6. The architecture of the Multi-Agent System . . . . .	35
5.2. Limitation . . . . .	35
5.3. Finite State Machine . . . . .	36
5.4. Group Tactic . . . . .	38
5.4.1. Target state prediction . . . . .	38
5.4.2. Focusing fire . . . . .	38
5.4.3. Encirclement and Group logic . . . . .	39
5.5. Structure of the implemented AI overview . . . . .	42
<b>6. Experiments and Results</b>	<b>43</b>
6.1. Experiment overview . . . . .	43
6.2. Conducted experiments . . . . .	46
6.3. Results of the experiment . . . . .	47
6.4. Discussion . . . . .	52
<b>7. Conclusion</b>	<b>53</b>
7.1. Further work . . . . .	54
<b>Bibliography</b>	<b>54</b>
<b>Appendix</b>	<b>I</b>
<b>A. CD content</b>	<b>II</b>

# List of Figures

3.1.	Structure of the JNIBWAPI wrapper . . . . .	12
4.1.	Basic Potential Fields examples [1] . . . . .	15
4.2.	Two potential fields with strength values shown . . . . .	16
4.3.	Generic finite state machine diagram . . . . .	18
4.4.	Fuzzy logic temperature [2] . . . . .	19
5.1.	Terrain field construction phases . . . . .	23
5.2.	Example of the Terrain field: (5.2a) Terrain obstacle and (5.2b) its Terrain Potential Field . . . . .	24
5.3.	The potential $PF_{static}$ generated by a dynamic object, in this case, building . . . . .	25
5.4.	Map view in the game . . . . .	26
5.5.	Potential field example . . . . .	26
5.6.	The Potential Function $PF_{aggressive}$ generated by enemy unit given the distance $D_t$ . . . . .	27
5.7.	The Potential Function $PF_{defensive}$ generated by enemy unit given the distance $D$ . . . . .	28
5.8.	Illustration of collision radiuses . . . . .	29
5.9.	The Potential Function $PF_{collision}$ generated by enemy unit given the distance $D$ . . . . .	30
5.10.	Construction of formation abscissa example . . . . .	31
5.11.	The Potential Function $PF_{formation}$ generated by enemy unit given the distance $D$ . . . . .	32
5.12.	The Potential Function $PF_{encirclement}$ generated by enemy unit given the distance $D$ . . . . .	33
5.13.	Illustration of Area Offset . . . . .	34
5.14.	Negative trail example . . . . .	35
5.15.	An illustration of the local optimal problem . . . . .	36
5.16.	State diagram for a agent . . . . .	37
5.17.	Illustration of the membership functions . . . . .	40
5.18.	Illustration of membership function . . . . .	41
5.19.	Illustration of membership functions . . . . .	41
6.1.	Example of the training maps, (6.1a) illustrates the general test training map. Figure (6.1b) illustrates modified test map . . . . .	44
6.2.	Military units used on training maps . . . . .	45
6.3.	Vizualization of the experiment results for opponent OAI . . . . .	49
6.4.	Vizualization of the experiment results for opponent OBAI . . . . .	50

6.5. Vizualization of the experiment results for opponent TSCAI . . . . . 51

# List of Tables

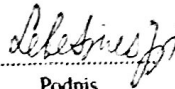
- 6.1. Overview of units properties . . . . . 45
- 6.2. Structure of the experiment . . . . . 46
- 6.3. Summary of the experiments for opponent OAI . . . . . 48
- 6.4. Summary of the experiments for opponent OBAI . . . . . 50
- 6.5. Summary of the experiments for opponent TSCAI . . . . . 51



### **Prohlášení**

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 7. 5. 2015

  
.....  
Podpis

# 1

## Chapter 1.

---

# Introduction

## 1.1. Motivation

Complex dynamic environments, where neither perfect nor complete information about the current state or about the dynamics of the environment are available, pose significant challenges for artificial intelligence (AI). Road traffic, finance, or weather forecasts are examples of such large, complex, real-life dynamic environments. The RTS games can be seen as a simplification of a one of a real-life environment, with simpler dynamics in a finite and smaller world with antagonistic environment, although still complex enough to study some of the key interesting problems like decision making (on several layers of abstraction e.g., tactical or strategic) under uncertainty, different aspects of control or real-time adversarial planning. Real-time strategy (RTS) is a genre of strategic game. Typical example of such games are Dune II [3], Command & Conquer [4], Warcraft II [5], Starcraft [6] and others. The precise definition of an RTS game has been debated for many years in the game industry. Blizzard's Rob Pardo has, for instance, defined an RTS game as: "A strategic game in which the primary mode of play is in a real-time setting" [7]. The concept of the game is based on a surviving in hostile environment, while a player has to build and manage facilities, create and control different types of units on a map with limited visibility range, and therefore, with uncompleted information. The goal is to defeat (usually it means elimination) enemy player who has similar start conditions. To do so, one should build and manage strong economy base (in the terms of the game defined resources), develop new technologies, train and upgrade army units. Generally such a game can be divided into the following subsections:

### Management of the Economy

- Different kinds of resources should be gathered. This process is performed by a worker type of the game unit. Flow of resources is usually limited by the

size of deposits site and not by the number of the workers who are assigned to gather the resources. Therefore, it is important to properly manage these units in order to build a strong economy that will allow to survive and defeat the enemies..

### **Management of the Buildings**

- The construction of new building is a complex task. There are few types of buildings: economy, technology, production, and defensive, all of them have different purpose, and therefore, they should be build on different places. Moreover, the buildings can be used as artificial barriers against enemy units, when they are properly placed, e.g., in narrow passages.

### **Recruitment of the new Units**

- There are many types of units and different types are suitable for different tasks. Where one unit can excel others may fail. The decision when and which type of unit to recruit is therefore very important.

### **Technology development**

- Investment into technology has long term benefits but it is usually expensive and slows down the player, especially at the beginning of the game, in a comparison to enemy which could rise large low-tech army and strike before the technology can outbalance the disadvantage. Due to this aspect of the game, careful decision making is needed.

### **Military unit control**

- The goal is the military victory. To achieve it, the player has to properly control his/her units and be able to defeat enemy army. The control and coordination of units are difficult tasks and a level of control determines how effective troops are. A correct use of unit roles (according to their types), proper positioning of the units and a fire support determine the result of encounter semi-equivalent armies (regarding strength of the groups), which could decide the whole game.

To provide challenging and entertaining game experience developers aim to create artificial intelligence that could handle all the tasks described above, because in a single player game, there are only computer players that influence the game experience. As we can see, RTS games are very complex problem, and therefore, AI is usually divided to several modules, each focused on the specific part of the required on-line decision-making process.

## 1.2. Problem Statement

In this thesis, we focused on the design of AI for movement control and coordination of the groups of mobile units. We will concentrate on the game called “Starcraft: Brood war” which, is very well known (released 1998 [6]), really spread (sold almost ten million copies combined [8]), the game is considered to be perfectly balanced, there are huge amount of expert level information sources (e.g., replays from tournaments) and due to this qualities there is a large community of players and actively updated application programming interfaces (APIs) for the game [9]. Additionally, StarCraft has been already used for studying AI aspects in real-time decision-making, for example [10, 11, 12].

## 1.3. Thesis Structure

This master thesis is structured into the following chapters each presenting different kinds of contributions for the project.

- Chapter 2: Background and related work, that has been done in the field of the thesis.
- Chapter 3: Setup of the test bed platform for the solution of the thesis.
- Chapter 4: Methodology of the approaches and algorithms used to achieve the goal of this thesis.
- Chapter 5: Detailed description of the methods used in the implementation of the AI.
- Chapter 6: Realized experiments and results , and found insights.
- Chapter 7: the conclusion of the work.

## Chapter 2.

---

# 2 Background and Related Work

In this chapter, we present an overview on variety of related research areas including description of specifics in RTS games, challenges in RTS Game AI, and existing works on RTS Game AI. The thesis is focused on applications to RTS games, specifically on works related to the game StarCraft.

### 2.1. Real-time strategy games

RTS games are games, where the main differences to the traditional games like Chess are:

- Actions and moves can be performed simultaneously by multiple players at the same time. Additionally, these actions are durative, i.e., actions are not instantaneous, but take some amount of time to complete. Also, after an execution of some action, there is a cool-down time. At this time, we cannot perform this particular action again.
- Real-time constriction means that there is a fixed amount of time to decide what the next moves can be. In comparison to the Chess, where players can take several minutes to decide the next step, in StarCraft, the game executes at the 24 frame per second, which means that player can act as fast as every 42 ms, before the game state changes [13].
- Most RTS games are partially observable which means that a players can have information about the map only for already explored places. This is referred as the fog-of-war. Additionally, the current information and the state of enemy units can be accessible only if the enemy units are in the range of sight of some allied units.
- Finally, the complexity of these games, both in terms of the state space size and the of number of actions available at each decision step, is very large. For example, the state space of the Chess to be typically estimated around  $10^{50}$ , heads up no-limit Texas holdem poker around  $10^{80}$ , and Go around  $10^{170}$ . In

comparison, the state space of the StarCraft in a typical map is estimated to  $10^{1685}$  [13].

## 2.2. StarCraft

StarCraft: Brood War is a popular RTS game released in 1998 by the Blizzard Entertainment. StarCraft has three distinct races that players can command: Protoss, Terran, and Zerg. Each race has unique units and technology graph (tech tree) which can be expanded to unlock new unit types, structures, and upgrades. Additionally, each of the race supports different styles of the gameplay. Mastering a single race requires a great deal of practice, and experts in this domain focus on playing a single race. One of the most remarkable aspects of StarCraf is that the three races are extremely well balanced. The races are:

- **Terrans**, the human-like race, that has versatile and flexible units, giving a balanced option between Protoss and Zergs.
- **Protoss**, the highly evolved humanoid race, it provides units with a long and expensive manufacturing process, but they are strong and resistant. These conditions make players follow a strategy of quality over quantity.
- **Zergs**, the insectoid race, it provides units that are cheap and weak. They can be produced fast and encourage players to overwhelm their opponents with sheer numbers.

In order to win a StarCraft game, players must firstly gather resources (minerals and Vespene gas). Players need to allocate that resources for creating more buildings (which reinforce the economy, and allows players to create units or unlock stronger units), research new technologies (in order to use new unit abilities or improve the units), and train military units. Units must be distributed to accomplish different tasks such as reconnaissance, defense, and attack. While performing all of those tasks, a player also needs to strategically understand the geometry of the map in order to decide where to place new buildings or where to set defensive outposts. Finally, when a player conducts offensive maneuver, each of the units in order to fight a battle has to take a proper position, which requires a quick and reactive control of each unit.

## 2.3. Challenges in RTS Game AI

There are several challenges in AI for RTS games [14]:

- Resource management
- Decision making under uncertainty
- Spatial and temporal reasoning
- Cooperation and collaboration(between multiple AIs)

- Opponent modeling and learning
- Adversarial real-time planning

Systems that play RTS games need to address most, if not all, the aforementioned problems together. Therefore, it is hard to classify existing work on RTS AI as particular approaches that address the different problems above. For that reason, we divide the approaches according to several levels of abstraction.

- Strategy corresponds to the high-level decision-making process. This is the highest level of abstraction for the game comprehension. Finding an efficient strategy or counter-strategy against a given opponent is the key problem in RTS games.
- Tactics are the implementations of the current strategies. It implies army and building positioning, movements, timing and etc. Tactics concern a group of units.
- Reactive control is the implementation of the tactics. This consists of moving, targeting, firing, fleeing, hit-and-run techniques (also known as “kiting”) during battle. Reactive control focuses on a specific (single) unit.
- Terrain analysis consists of the analysis of regions and composing the map: choke-points, minerals and gas emplacement, low and high walkable grounds, islands, etc.
- Intelligence gathering corresponds to information collected about the opponent. Because of the fog-of-war, players must regularly send scouts to localize and spy enemy bases.

## 2.4. Existing work on RTS Game AI

In this thesis, we focus on area of Tactics and Reactive control (in the terms of abstraction defined above); hence, state of art in this section is also focused on that aspects of AI.

### 2.4.1. Tactics

The tactical reasoning involves reasoning about different abilities in a group and about the terrain and positions of the different groups of units in order to gain a military advantage in battles. We divide the work about tactical reasoning into two main parts: terrain analysis and decision-making. The terrain analysis provides the AI with structured information about the map in order to help making decisions. This analysis is usually performed off-line, in order to save CPU time during the game. In the most of the state of arts approaches, StarCraft bots use the BWTA library<sup>1</sup>. This library is based on the work of Prekins et al. [15] who applied

---

<sup>1</sup><https://code.google.com/p/bwta/>

Voronoi decomposition and the necessary pruning of leaves (then pruning) to detect regions and relevant choke points in RTS maps. Concerning tactical decision-making, many different approaches have been explored in the field of AI for RTS games such as machine learning or game tree search. Sharma et al. [16] combined Case Based Reasoning (CBR) and reinforcement learning to enable reuse of tactical plan components. Cadena and Garrido [17] used fuzzy CBR for strategic and tactical planning. Finally, Miles [18] proposed an idea of the Influence Map Trees (IMTrees), the tree where each leaf node is an influence map and each intermediate node is a combination operation (sum, multiplication). Miles used evolutionary algorithms to learn IMTrees for each strategic decision in the game involving spatial reasoning by combining a set of basic influence maps. Game tree search techniques have been explored for tactical decision-making. Churchill and Bruno [19] presented the Alpha-Beta Considering Durations (ABCD) algorithm, a game tree search algorithm for tactical battles in the RTS games. To make game tree search applicable at this level, abstract game state representations are used in order to reduce the complexity.

## 2.4.2. Reactive control

The reactive control aims at maximize the effectivity of units, including simultaneous control of units of different types in complex battles on heterogeneous terrain. Since the core of this thesis falls right in this thread, we discuss it more detaily.

### Finite State Machines

Although Finite State Machines (FSM) have been around for a long time, they are still quite common and useful in modern games. The fact, they are relatively easy to understand, implement, and to debug which are the main aspects that contribute to their frequent usage in the game development. An example of the application of the FSM in games is presented by Wenfeng Hu et al. work [20]. Downsides are a large number of states (for the treatment of all possible situations), purely FSM solutions are hard to maintain and modify. Besides, FSM solutions are also often inflexible and predictable. Hierarchical FSM (HFSM) solves some of these problems and evolves into behavior trees (BT, hybrids HFSM) [21]. However, adaptivity of the behavior by parameters learning is not the main focus of these models.

### Searching tree or graph data structures

Heuristic search has been very successful in abstract game domains such as Chess and Go. A general approach is to construct game space (weighted graph) from units and available actions and then execute some kind of heuristic search, e.g., min-max, alpha-beta search or other. A similar approach has been applied by



Churchill and Bruno [19]. They present a variant of the alpha-beta search capable of dealing with simultaneous moves and durative actions. The method yields a promising results. On the other hand, the algorithm has very high computational and memory requirements. The high complexity of the game space and hard time restriction on computations runtime significantly limits the useability of this method in full-scale game AI. In the work mentioned above, experiments were limited to small scale combat scenarios (8 vs. 8 units).

Another example of a similar approach, that is worth mentioning, is the Monte Carlo search. Wang Zhe et al. [22] present Monte-Carlo Planning AI. This work solves some problem associated with the computational cost of the previous methods, unfortunately, it is very sensitive to accurate game simulation which is used to evaluate moves.

### **Flock and Swarm intelligence**

This kind of intelligence is an emergent behavior that some natural systems exhibit, such as ant colonies, flock of birds, swarms of bees, etc. There are efforts to apply these mechanisms to RTS games. Ivan Gonzalez and Leonardo Garrido [12] proposed reactive control technique based on swarm behavior, which subsumes spatial distribution and navigation. The method is scalable and robust. However, this approach is intended for homogenous group of units, which might not be fulfilled in RTS games due to large variety of unit types.

### **Potential fields and influence maps**

Numerous studies concerning potential fields are related to spatial navigation, obstacle avoidance, and units navigation, see [23, 24]. The idea behind this technique is the imitation of conservative natural force fields, which can be derived as the gradient of a potential function. A value of this function is proportional to the force created by the potential field at a given point in the space. If we have planar space represented as a grid, which is our case, we can create 2D array, where  $(i, j)$  coordinates correspond to the actual position  $(x, y)$  of the object and the value of the element in the array corresponds to potential function value in that point. This array is called influence map. Johan Hagelbäck [10] presented a Multi-Agent Potential Fields based bot that is able to successfully integrate reactive control, navigation, and exploration. Danielsiek et al. [25] used influence maps combined with the flocking technique to achieve an intelligent squad movement and ability to flank enemy squad. Despite their success, drawbacks of potential field-based techniques are:

- Local optima problem. Unit controlled by the potential field usually does not have any path finding. Hence, units can stuck inside a U-shaped obstacle. This problem is described in detail in Section 4.1.
- Performance issues. To calculate how multiple fields affect all positions in

the game map, is computationally demanding (for run-time calculation) or a lot of memory is required (if the potential field is pre-calculated and stored).

- Parameters tuning. In potential field based solution there are typically multiple fields. Each field is aimed on different aspect of the control, such as collision avoidance, navigation etc. The contributions of each potential field are not the same, therefore the fields have to be weighted. To achieve desired behavior of the agent, particular weights have to be tuned. This can be done by tuning weights manually or by auto learning technique. Approaches for automatic learning by tuning of weights have been explored, for example using reinforcement learning [26] or genetic algorithm tuning [27].

### **Potential flow**

The potential flow is a concept originated from fluid dynamics and it is defined as inviscid, irrotational flow of ideal fluid. A two-dimensional flow is represented by a complex, harmonic function. The advantages over a regular potential field are that the harmonic function is a solution of the Laplace equation, and therefore, it does not have local minima. The property of the potential flow satisfies the principle of superposition, which means that various potential flows can be added together to generate a new complicated potential flows. In the work of Tung Nguyen et al. [28], the potential flow is used for unit positioning during a combat. The experimental results show a promising potential and they are clearly superior to commonly used method.

### **Bayesian modeling**

Gabriel Synnaeve and Pierre Bessiere [11] presented Bayesian programs, which can be considered as a formalism that to describe entirely any kind of Bayesian model. Bayesian model is a type of statistical model that represents a set of random variables and their conditional dependencies via likelihood functions and prior probability functions. This approach subsumes finite state machine and potential fields, allowing for an integration of tactical goals directly in a reactive control level.

### **Reinforcement learning**

Reinforcement learning is an unsupervised machine learning technique, which puts an agent into an unknown environment, giving it a set of possible actions with the aim to maximize a reward. Wender and Watson [29] evaluated Q-learning and Sarsa algorithms [30] for decentralized reactive control. The downside of this approach is a relatively long learning time. There are efforts to accelerate the learning process, for example Jaide and Munoz-Avila [31] used learning through just one Q-function for each unit type in order to reduce the search space.

## **Cognitive control**

Finally, a reactive control research has been performed in the field of cognitive science, where Wintermute et al. [32] have explored AI human-like behavior in RTS games. The middleware constructed based on cognitive architecture supports grouping, attention, coordinated path finding, and FSM control of low-level units behaviors. Since the aim of the approach is not to outperform other AIs but rather to make the behavior of this AI more similar to human gameplay it is possible to say the goal has been achieved. This approach emulates the adaptation, anticipation, and estimation capabilities demonstrated by human players.

## Chapter 3.

# 3 Platform Setup

---

This chapter describes the software and the hardware used as the platform for implementation of the objective of the thesis.

### 3.1. StarCraft

As it was mentioned earlier, the RTS game StarCraft: Brood War is used as test platform for this thesis. Since StarCraft is not open-source (can be purchased here<sup>1</sup>), the free open-source framework called Brood War Application Programming Interface (BWAPI) has been used to communicate with the game.

### 3.2. BWAPI

BWAPI is written in C++, but several different wrappers for the other programming languages are created including Java, Python, PHP, Ruby and others. I have chosen the Java language and hence Java wrapper JNIBWAPI. JNIBWAPI uses Java Native Interface [33] to communicate between Java and BWAPI's C++ code. JNIBWAPI itself works as a BWAPI AI Client, with the C++ code acting as a binding between the BWAPI and Java processes. By implementing the BWAPIEventListener interface, a Java class can receive callbacks identical to a BWAPI AI Module. The JNI interface provides several advantages over the previous socket-based ProxyBot interface. First, the interface uses the BWAPI shared memory bridge which lessens the communication bottleneck between C++ and Java and prevents cheating using a direct memory access. Second, the BWAPI utility functions (e.g., canBuildHere) can now be called from Java. Finally, the use of JNI should result in easier project maintainability and extensibility.

---

<sup>1</sup> <https://eu.battle.net/shop/en/?id=110000124>

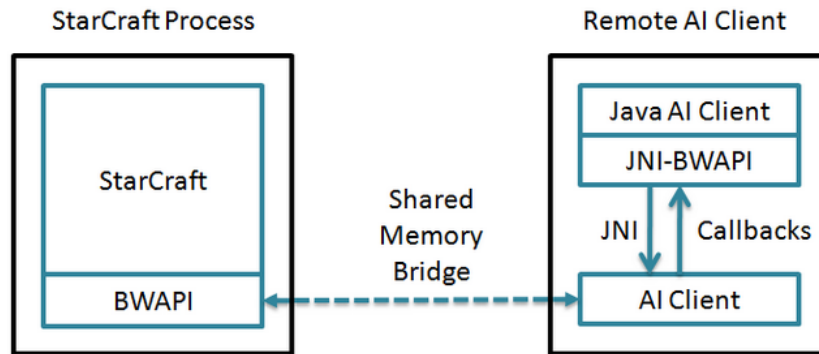


Figure 3.1.: Structure of the JNIBWAPI wrapper

### 3.2.1. Physical Properties

A StarCraft map is a two-dimensional, tile-based map. The size of a StarCraft map can vary between  $64 \times 64$  up to  $256 \times 256$  build tiles. There are three different types of coordinates for a StarCraft map also included in JNIBWAPI:

- Pixel Tiles, which are measured in  $1 \times 1$  pixels and have the highest resolution.
- Walk Tiles, which are measured in  $8 \times 8$  pixels and can be used for units, when walking around on a map (walkability data is available at this resolution).
- Build Tiles, which are measured in  $32 \times 32$  pixels and are useful for build management (buildability data is available at this resolution).

### 3.2.2. Unit control methods

In order to be able to control and give individual units orders, JNIBWAPI includes the following applicable methods:

#### **move()**

Takes a target position as input, which refers to the place where a specific unit is ordered to move.

#### **rightClick()**

The `rightClick()` method works like the right click in the GUI. The method is equivalent to `move()` method in a way, but it has a lower CPU execution time, hence the `rightClick()` method is used in this thesis. There is also an overloaded `rightClick()` version which takes unit as the input. This is equivalent to in game attack order if the unit is hostile.

**attackUnit()**

Takes a target enemy unit as the input and moves toward it until it is within max shooting distance, then it starts to attack.

**stop()**

Order the unit to abort any activity.

### 3.3. System Requirements

Below are listed all programs and utilities required for running our develop AI for the StarCraft RTS game.

- StarCraft:Brood War version 1.16.1<sup>2</sup>
- ChaosLauncher version 0.5.4.4<sup>3</sup>, additionally ChaosLauncher plugins: BWAPI Injector (1.16.1.) RELEASE and W-MODE 1.02
- BWAPI version 3.7.4<sup>4</sup>
- JNIBWAPI version 1.0<sup>5</sup>
- 32 bit JRE (Java Runtime Environment) version 7u76 or higher<sup>6</sup>
- Eclipse IDE for Java developers<sup>7</sup>
- Microsoft Windows 7-BWAPI uses dll injection and it is therefore unlikely to work in Wine or Mac<sup>8</sup>

---

<sup>2</sup> <http://eu.blizzard.com/en-gb/games/sc/>

<sup>3</sup> <http://winner.cspsx.de/Starcraft/>

<sup>4</sup> <https://code.google.com/p/bwapi/>

<sup>5</sup> <https://github.com/JNIBWAPI/JNIBWAPI>

<sup>6</sup> <http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

<sup>7</sup> <https://www.eclipse.org/downloads/>

<sup>8</sup> <http://eis.ucsc.edu/StarCraftBWAPI>

# 4

## Chapter 4.

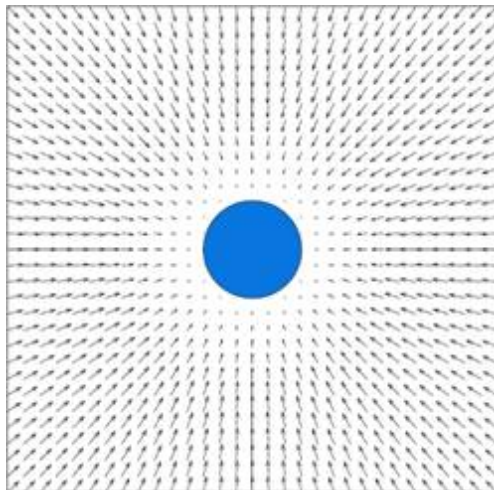
---

# Methodology

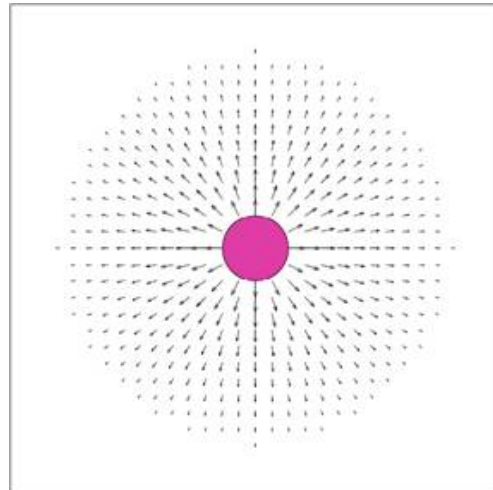
This chapter describes the techniques and algorithms used to achieve the objective of this thesis (the design of AI for movement control and coordination of a groups of mobile units), including Potential Fields, Multi-Agent Potential Fields and Bayesian function.

### 4.1. Potential Fields

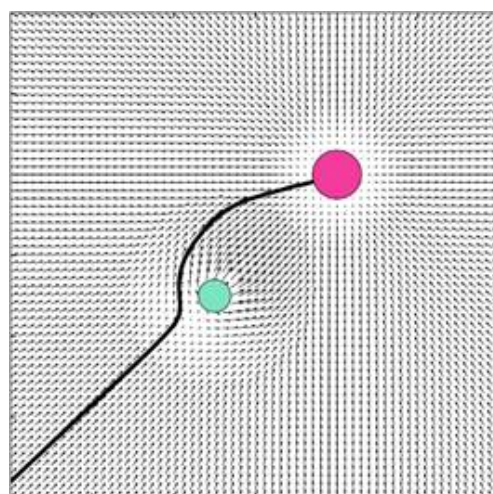
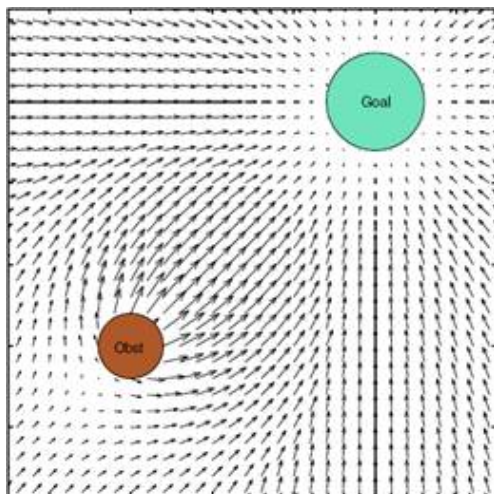
A concept of Potential fields (PF) originates from robotics. It was firstly introduced by Khatib [34] for a real-time obstacle avoidance ,in planning and control of manipulators and mobile robots. The principle of the concept is based on placing, attracting or repelling charges at important locations in a virtual world. An attracting charge is placed at the position to be reached, and repelling charges are placed at the positions of obstacles. They can be thought off as magnetic charges working on a charged particle. The force (attractive or repulsive) from a PF decreases with the distance from the center of the position of the charge. It can decrease in different ways: linearly, exponentially or discretely. The different variations represent different desired behaviors. The Figure 4.1 shows an example of basic PF application.



(a) A attractive Potential Field



(b) A repelling Potential Field



(c) The two behaviors, seeking and avoiding, can be combined by combining the two potential fields, the agent then can follow the force induced by the new field to reach the goal while avoiding the obstacle

Figure 4.1.: Basic Potential Fields examples [1]

Each charge generates a field of a specific size. The different fields are weighted and summed together to form an aggregated field. The total field can be used for navigation by letting the robot move to the most attracting position in its near surroundings [35]. The potential field can be represented as the Influence map. The idea is to put a charge at an interesting position in the game world and let the charge generate a field that gradually fades to zero. The charge can be attracting (positive) or repelling (negative). An example of how this could look like is shown in Figure 4.2. In this thesis, potential fields are always represented as the Influence map.



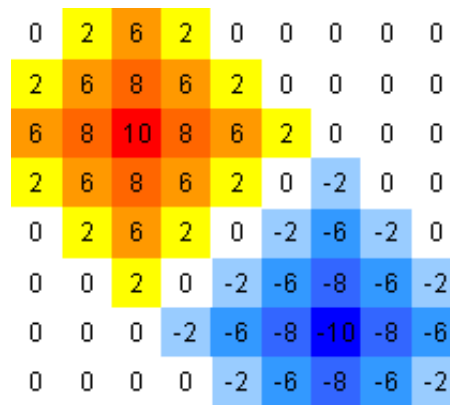


Figure 4.2.: Two potential fields with strength values shown

In the movement control method based on PF, the agent looks only one step ahead to check the 8 surrounding tiles. Then, it uses the tile with the highest attractive value, proceed to move to it. If the highest returned value is the tile where the agent is currently placed, the agent will stand idle. This status of agent can mean that the agent has reached its target position or that the agent has been stuck in a local optima. This issue is the typical problem of this approach. There are several proposed solutions. Thuraul et al. [36] has described a solution which lets the agents generate a negative trail on a specified number of previous visited tiles, which is similar to a pheromone trail in the ant colony optimization algorithm, e.g., [37]. The trail will depreciate the evaluation of the visited positions, and therefore it pushes the agent away from the local optima. Notice, that if highly evaluated position is the global optima, the agent is forced to return. This is explained more in Section 5.2.

There is obviously the question of granularity. The value from each PF can be assigned to each pixel or let several pixels represent one cell in a grid. This is actually a good way to tune computational power needed [10]. One does not have to represent such a grid explicitly. It is possible to use some function that is dependent on the position (for example use the distance between the agent and the fields) to calculate values of the PF and create a result vector. This approach is often used in the Multi-Agent Potential Fields method. If multiple units are using the same fields, e.g., if the fields are set by a centralized intelligence, representing them in a grid is reasonable. The grid might be calculated beforehand and each unit might only need to check the values of the spaces closest to them. If each unit has its own PFs, calculating them this way would be impractical, as it could means that each unit could have to represent the whole grid and then decide what to do. Using the Multi-Agent Potential Fields method seems to be the best choice. Therefore it is detaily described in the next section.

## 4.2. Multi-Agent Potential Fields

Multi-Agent Potential Fields (MAPF) for RTS games were firstly introduced by Hagelback and Johansson [38], where a methodology for how to design a MAPF-based solution for an RTS game is described as follows:

1. **The identification of objects.** The purpose of this phase is to identify all objects that will effect or have some kind of impact on the agents. These objects can be either static (which means they will always remain unchanged), or dynamic, meaning that they can move around, change shape or change their type of impact (repulsive to attractive or vice versa) on the agents.
2. **The identification of the driving forces (fields) of the game.** The purpose of this phase is to identify the different force fields, which will be placed on tactical positions in the game world, to push the agents towards their overall goal. These fields can be repulsive forces generated by e.g. static terrain objects, attractive forces generated by for example the goal object (such as nearest enemy spotted unit or base).
3. **The process of assigning charges to the objects.** The purpose of this phase is to specify the formulas for calculating each specific potential field value. High potential field values will attract the agents, while low potential field values will repulse the agents.
4. **The granularity of time and space in the environment.** The purpose of this phase is to decide on the resolution of time and space. Resolution of space is a measure for the tile size in the grid of potential fields. The resolution of time is a measure for how far the agents can get in a game frame. In other words, how far ahead agent should look.
5. **The agents of the system.** The purpose of this phase is to identify the objects that should become agents controlled by the AI, and decide which actions these agents should perform in different states, based on the total potential field values in the surrounding tiles.
6. **The architecture of the Multi-Agent System.** The purpose of the last phase is to decide what the high level strategies should be. The different types of agents should have different roles (not covered by a basic unit agent).

## 4.3. Bayesian function

For the evaluation of multiple PFs in Multi-Agent System there is a need for one overall PF for each agent, which will be used by the agent when choosing its next actions. The contributions of each potential field are not the same, therefore the fields are weighted. To achieve desired behavior, weights have to be tuned. In our case we tune weights manually. For simplicity, all PFs are calculated in normalized form and then multiplied by weights. Hence, all fields are calculated in

the same range of values, specifically in the range  $\langle 0, 1 \rangle$ . This approach gives us possibility to calculate normalized overall PF without weights and tune individual contributions later. Additionally, we can see these PFs as probabilities in a joint distribution. The joint distribution constructed based on this principle is called inverse programming [39]. The sensory variables (stored as a PF) are considered independent knowing the actions, contrary to standard naive Bayesian fusion, in which the sensory variables are considered independent knowing the phenomenon,

$$P(Pos_{x,y}, PF_1, PF_2, \dots, PF_n) = P(Pos_{x,y}) \cdot \prod_{i=1}^n P(Pos_{x,y}|PF_i) \quad (4.1)$$

where  $Pos_{x,y}$  is the position in terms of the game grid on the coordinates  $(x, y)$ ,  $PF_i$  is the contribution from the given PF,  $P(Pos_{x,y}|PF_i)$  is the conditional probability. For example,  $P(Pos_{x,y}|PF_{dmg})$  corresponds to a probability of the damage values on  $Pos_{x,y}$  and  $P(Pos_{x,y})$  is a prior probability (used uniform over all positions,  $P(Pos_{x,y}) = \frac{1}{m}$ , where  $m$  is the number of evaluated positions for the given agent). This joint distribution is used as the overall PF in the Multi-Agent System. One of the preferred features of this approach is that every PF has a possibility to set an evaluation of the given position to zero. This practically means that this position is forbidden to an agent regardless other fields values. For example, an agent is restricted to go to highly rated position when there is an obstacle on the same position.

## 4.4. Finite State Machine

A Finite State Machine(FSM) is an abstract machine that can exist in one of several different and predefined states. A finite state machine also can define a set of conditions that determine when the state should change. The actual state determines how the state machine behaves. This allows us to switch between agent roles based on unit state (the current Hit Points, reload timer, etc.) and unit type. The diagram in the Figure 4.3 illustrates how a simple finite state machine can be modeled.

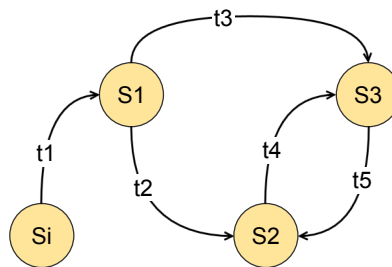


Figure 4.3.: Generic finite state machine diagram

In Figure 4.3, each potential state is illustrated as an orange disc. There are four possible states  $S_i, S_1, S_2, S_3$ . Naturally, every finite state machine also needs to move from one state to another state. In this case, the transition functions are illustrated as  $t_1, t_2, t_3, t_4, t_5$ . The finite state machine starts at the initial state  $S_i$ . It remains in this state until the  $t_1$  transition function provides a stimulus. Once the stimulus is provided, the state switches to  $S_1$ . The process repeats itself analogously for all states.

## 4.5. Fuzzy logic

In 1965, Lotfi Zadeh, a professor at the University of California Berkeley, wrote his original paper about fuzzy set theory. In 1994, an interview of Zadeh conducted by Jack Woehr of Dr. Dobbs Journal, Woehr paraphrases Zadeh when he says “fuzzy logic is a means of presenting problems to computers in a way akin to the way humans solve them”. Zadeh later goes on to say that “the essence of fuzzy logic is that everything is a matter of degree.” Fuzzy logic is used in a wide variety of real-world control applications such as controlling trains, air conditioning and heating systems, and robots, among other applications. Video games also offer many opportunities for fuzzy control [40]. The fuzzy control or inference process comprises three basic steps: fuzzyfication, evaluation of fuzzy rules and defuzzyfication. In this, thesis only fuzzy output is used; hence we consider that the defuzzyfication is not needed to be explained.

### 4.5.1. Fuzzyfication

An input to a fuzzy system can originate in the form of the crisp numbers. These numbers are real quantifying the input variables. The membership functions map input variables to a degree of membership, in a fuzzy set, between 0 and 1. If the degree of membership in a given set is 1, then we can say the input for that set is absolutely true. If the degree is 0, then we can say for that set the input is absolutely false. If the degree is between 0 and 1, it is true to a certain extent—that is, to a degree. An example of membership function used for fuzzyfication is shown in Figure 4.4.

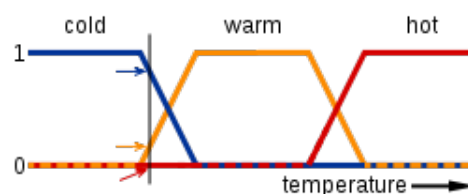


Figure 4.4.: Fuzzy logic temperature [2]

### 4.5.2. Evaluation of fuzzy rules

Once we expressed all the inputs to the system in terms of fuzzy set membership, we can combine them using fuzzy rules to determine the degree to which each rule is true. In other words, we can find the strength of each rule or the degree of membership in an output or action. Axioms for evaluation of fuzzy variables are given as:

$$\text{Truth}((A) \text{ OR } (B)) = \max(\text{Truth}(A), \text{Truth}(B)) \quad (4.2)$$

$$\text{Truth}((A) \text{ AND } (B)) = \min(\text{Truth}(A), \text{Truth}(B)) \quad (4.3)$$

$$\text{Truth}(\text{NOT } (A)) = 1 - \text{Truth}(A) \quad (4.4)$$

Here, again, Truth(A) means the degree of membership of A in some fuzzy set. This will be a real number between 0 and 1. The same applies for Truth(B) as well.

# 5 Chapter 5.

---

## Agent Implementation

This chapter provides a description of the design and implementation of the proposed Multi-Agent Potential Field based solution using the 6 phases of the original methodology proposed [38] that has been extended for an additional agent logic and group logic.

### 5.1. Multi-Agent Potential Fields

#### 5.1.1. Identifying object

There are various objects in StarCraft. Basic identification and classification can be divided into two categories, namely the static objects and dynamic objects. Static objects includes terrain obstacles, like map edges, mountains, walls and static neutral units, like minerals and vespene geyser fields. Dynamic object category includes player's units, buildings and neutral units. Note that buildings are dynamic objects despite of their static character. The reason is that buildings can be destroyed or some of them can be even moved.

#### 5.1.2. Identifying fields

In this thesis, I have identified 7 different potential fields used in Multi-Agent Potential Field agent based on. PFs are described as follows:

1. Terrain field. This PF is generated by repelling static terrain. Purpose of this field is that agents can avoid getting too close to positions where they may get stuck or been pushed into a corner. Due to the fact that objects related to this field are time invariant, this field is pre-computed and stored in the memory.
2. Static field. This PF is an extension of the Terrain field. In this field, the repulsion forces from static neutral units and buildings, are subsumed with the terrain field. As mentioned above, buildings and static neutral units like minerals, etc. are slowly varying objects, hence this field is updated less frequently.

3. Aggressive field. This field is generated by enemy units. It is an attractive field that makes agents go towards the enemy units and place themselves in an appropriate distance, where they can fight with the enemies. Additionally, when an agent is a range unit (a unit with an abilities to make ranged attacks), small repelling field is placed on enemy units to avoid placing themselves unnecessary close to enemies.
4. Defensive field. This field is similar to the Aggressive field. It is also generated by enemy units. The difference is that repelling field is generated instead of attractive field. It makes agent to run away from enemies.
5. Collision field. All dynamic objects described above generate this kind of field that is repeling. The purpose is that agents can avoid colliding with each other as well as avoiding other units including enemies and neutral units.
6. Formation field. This field is used to attract the squad units to stay together and to minimize the risk of letting the AI's own units getting caught alone by an enemy. At the same time, this field works as a navigation field that leads agents to a goal of the squad. Finally, this gives us the opportunity to shape the formation in which the unit arrives to the enemy, which works as a pre-placement for units in squads.
7. Encirclement field. Basically, it is a kind of field that has the same structure as the Aggressive field. It has attractive and repelling component. The main difference is that the attraction radius of the field is significantly larger. Namely it has the size of the sight range of the enemy units. It enables us to surround a group of enemies without seeing our group by the enemy.

### 5.1.3. Assigning charges to the objects

Each object has a set of charges which generates a potential field around the object. Below, a more detailed description of the different fields is presented. All fields generated by objects are weighted and overall PF is created using (4.1). It is used by agents during the selections of the action.

#### Terrain field

The terrain obstacles generate a repelling field to support collision avoidance. The maximal charges are assigned to every terrain object. To assure the avoidance of edges is as smooth as possible, the potential field is smoothed by linear smoothing filter [41, 42]. This method removes noise (edges) by convolution of the original field with a mask that represents a low-pass filter. This convolution brings us the value of each pixel into closer harmony with the values of its neighbors. Basically, the smoothing filter sets each pixel to the average value of itself and its nearby neighbors. Finally, the field with charges and the smoothed field are compared. The highest value of each cell is selected as a value for the cell in the final Terrain field. Note, that inverse logic is used in the construction of the Terrain field. Instead

of assigning minimal charges (zero) to obstacles and filling the rest of the field with the maximal charges (one), we use inverted charges and as a evaluation function we use the following function (5.1).

$$P(Pos_{x,y}|PF_{terrain}) = 1 - PF_{terrain}(D) \quad (5.1)$$

Illustration of the construction steps are shown in Figure 5.1,

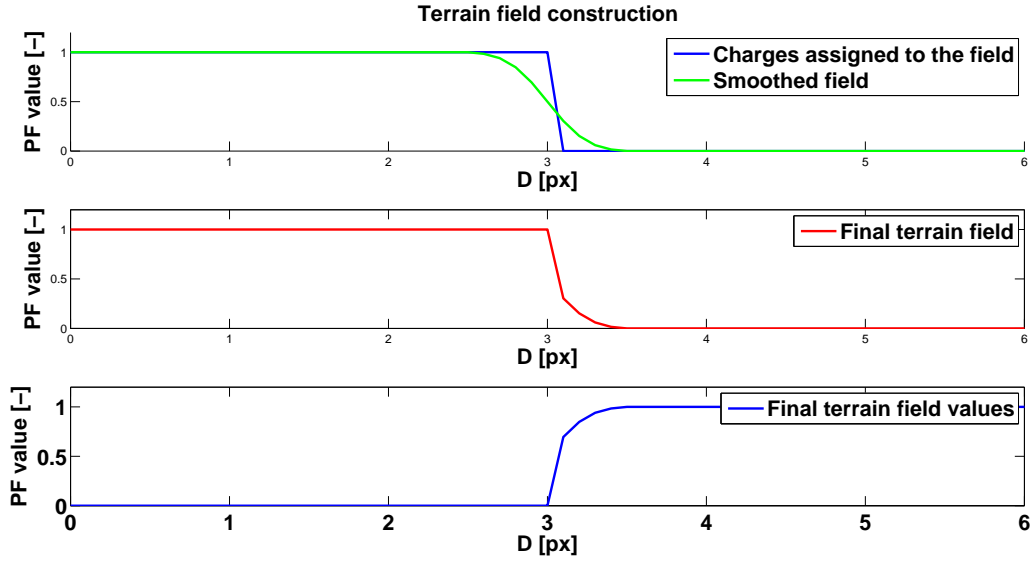
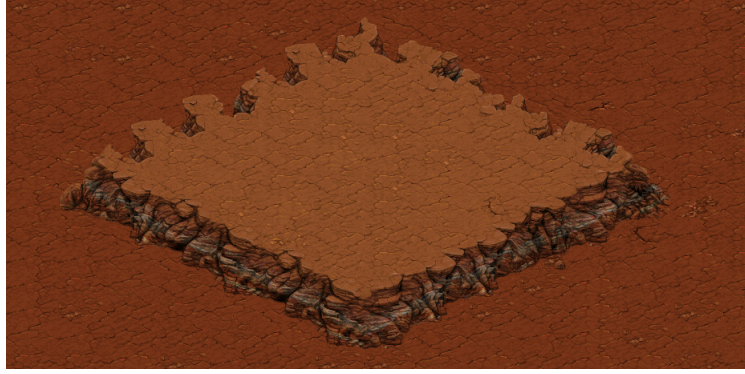


Figure 5.1.: Terrain field construction phases

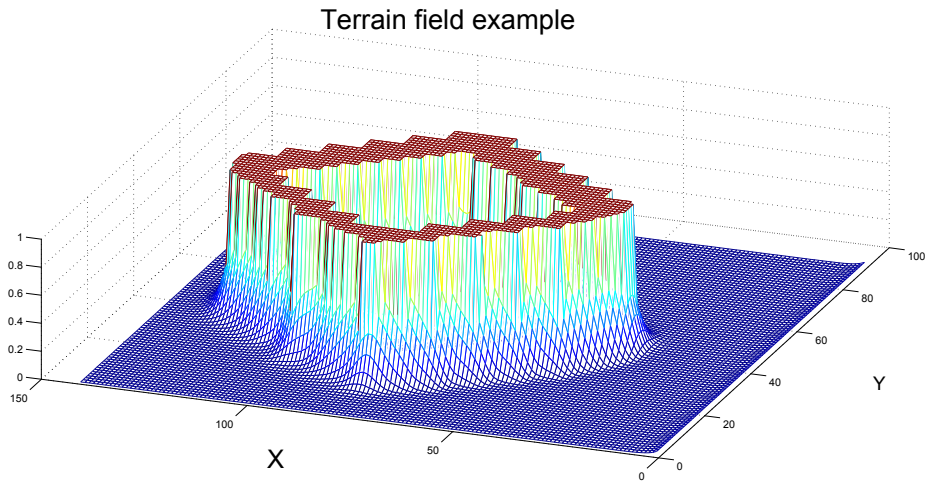
where  $D$  is a distance from an obstacle measured in pixels.

The representation of the in-game terrain obstacles as the Terrain field is shown in Figure 5.2.





(a)



(b)

Figure 5.2.: Example of the Terrain field: (5.2a) Terrain obstacle and (5.2b) its Terrain Potential Field

### Static field

Dynamic objects that change in time slowly (most of them can not change its position, but they only can disappear or be destroyed) like building and static neutral units, also generate a repelling field for collision avoidance. The Static field an example is depicted in Figure 5.3) at the distance  $D$  (in pixels) from the center of a object is calculated as:

$$PF_{static}(D) = \begin{cases} 1 & \text{if } D \leq OS \\ 0.8 & \text{if } (D + 2 \cdot 8) \leq OS \\ 0.5 & \text{if } (D + 4 \cdot 8) \leq OS \end{cases}, \quad (5.2)$$

where  $OS$  is an object size, defined as the distance from the object center to the boundary of the object shape.

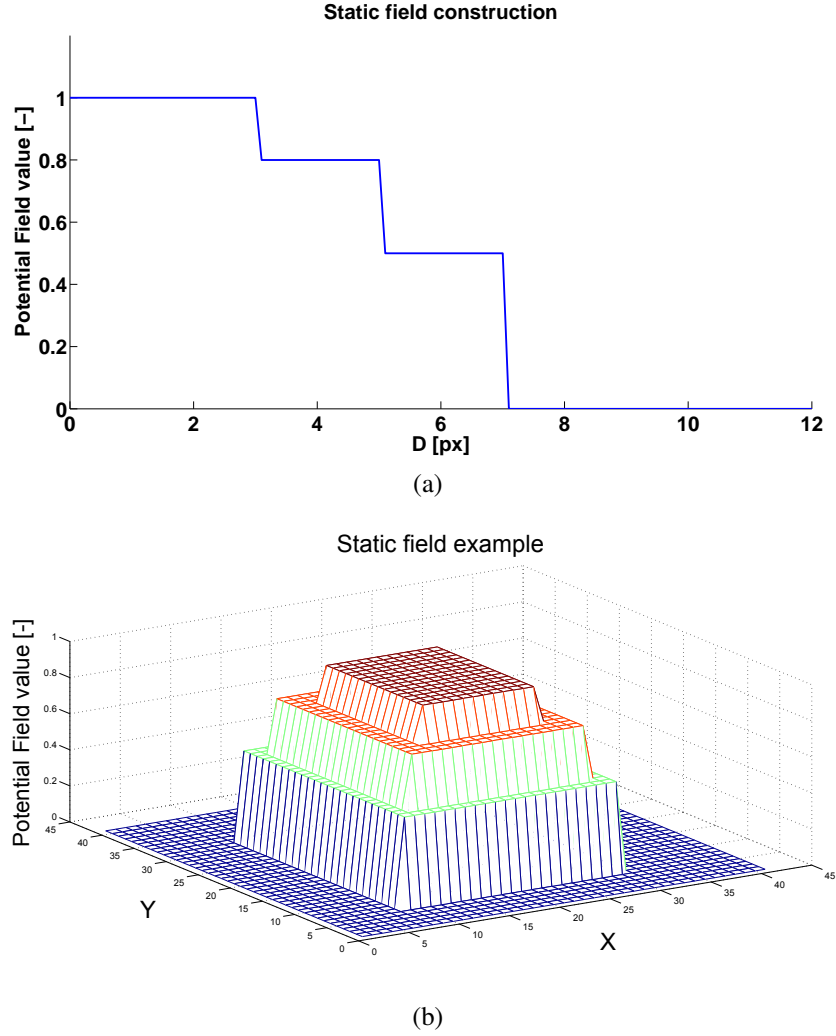


Figure 5.3.: The potential  $PF_{static}$  generated by a dynamic object, in this case, building

As in the Terrain field, the evaluation function is calculated as the inverted Static field value:

$$P(Pos_{x,y}|PF_{static}) = 1 - PF_{static}(D). \quad (5.3)$$

**Note:**

The Terrain and the Static fields are globally used PFs. Therefore, these fields are not MAPFs and they are different from all other PFs that are presented in this thesis. These fields are pre-computed and stored in a memory. As it was mentioned above, the Terrain field is pre-computed during initialization of the game and stored as an two-dimensional field. The Static field is computed in a case that there is a change in the state of static objects. For example the Static field update is performed if the number of objects, related to the field, is changed. Every agent can access to these fields and find required data related to its position. Example of these fields

subsumed into one Potential Field is shown in Figure 5.5. Notice, the map edges are considered as terrain obstacles.

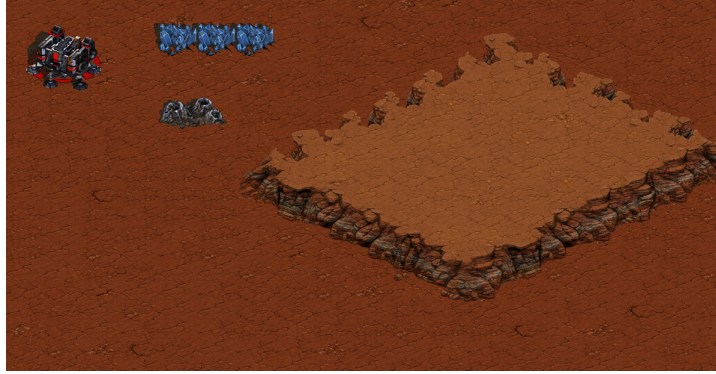


Figure 5.4.: Map view in the game

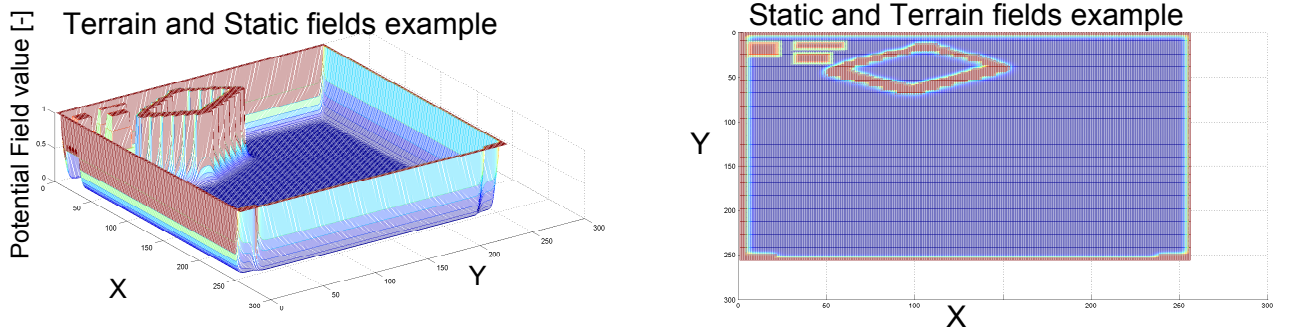


Figure 5.5.: Potential field example

### Aggressive field

All enemy units generate a symmetric surrounding field. The highest potential of the field is in the ring around the unit with a radius of the agent Maximum Shooting Distance (MSD). The Aggressive field (depicted in Figure 5.6) at the distance  $D$  (in pixels) from the center of the enemy unit is calculated as:

$$PF_{aggressive} = \begin{cases} \left(1 - \frac{D}{MD_{max}}\right)^2 & \text{if } D \geq MSD \\ \left(\frac{D}{MD_{max}}\right)^4 & \text{if } D < MSD \end{cases}, \quad (5.4)$$

where  $MD_{max}$  is the maximal possible distance in the given map, which corresponds to the length of its size diagonal line.

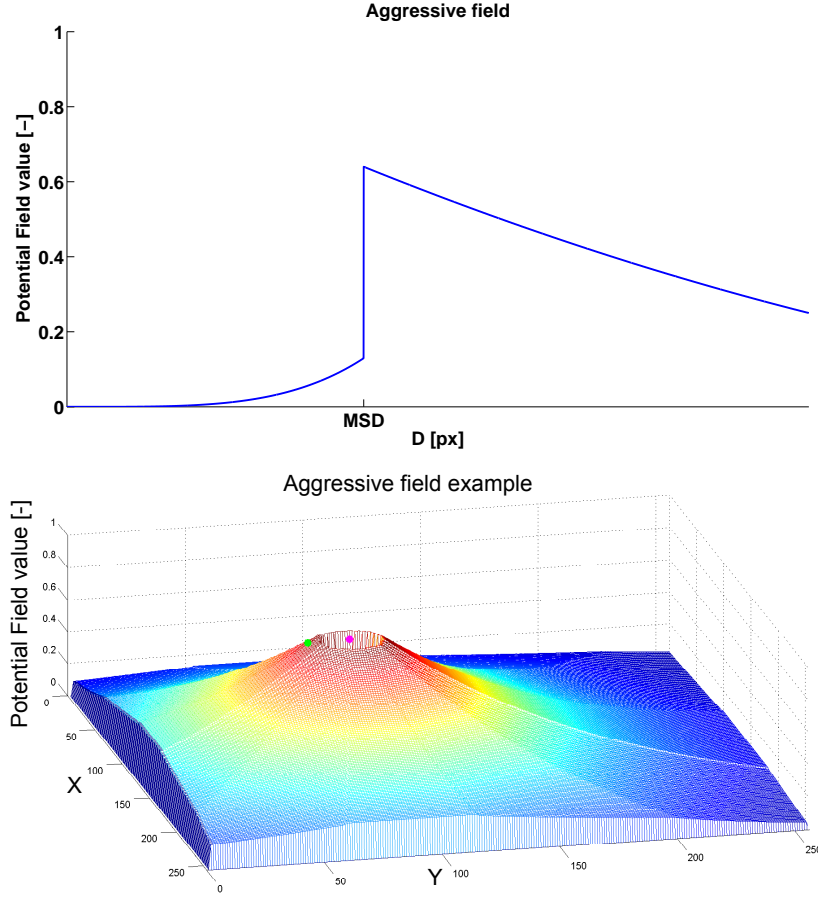


Figure 5.6.: The Potential Function  $PF_{aggressive}$  generated by enemy unit given the distance  $D$  t

Evaluation function is calculated as:

$$P(Pos_{x,y}|PF_{aggressive}) = PF_{aggressive}(D). \quad (5.5)$$

### Defensive field

All enemy units generate a symmetric surrounding field in the shape of a ring around the object (enemy unit) with the radius  $R$ . Generated repelling field should ensure, that agents keep a safe distance from enemy units. The radius  $R$  is calculated as

$$R = \max(eMSD, MSD) + 6 \cdot 8, \quad (5.6)$$

where  $eMSD$  is  $MSD$  of the enemy units. This field can be used for damage avoidance (therefore  $eMSD$  is incorporated in  $R$ ) or kiting (therefore  $MSD$  is incorporated in  $R$ ). Kiting is a reactive control tactic, which is further described in Section 5.1.5.

Defensive field (depicted in Figure 5.7) at the distance  $D$  (in pixels) from the center of the enemy unit is calculated as:

$$PF_{defensive} = \begin{cases} 0 & \text{if } D > R \\ (1 - \frac{D}{R})^2 & \text{if } D \leq R \end{cases} \quad (5.7)$$

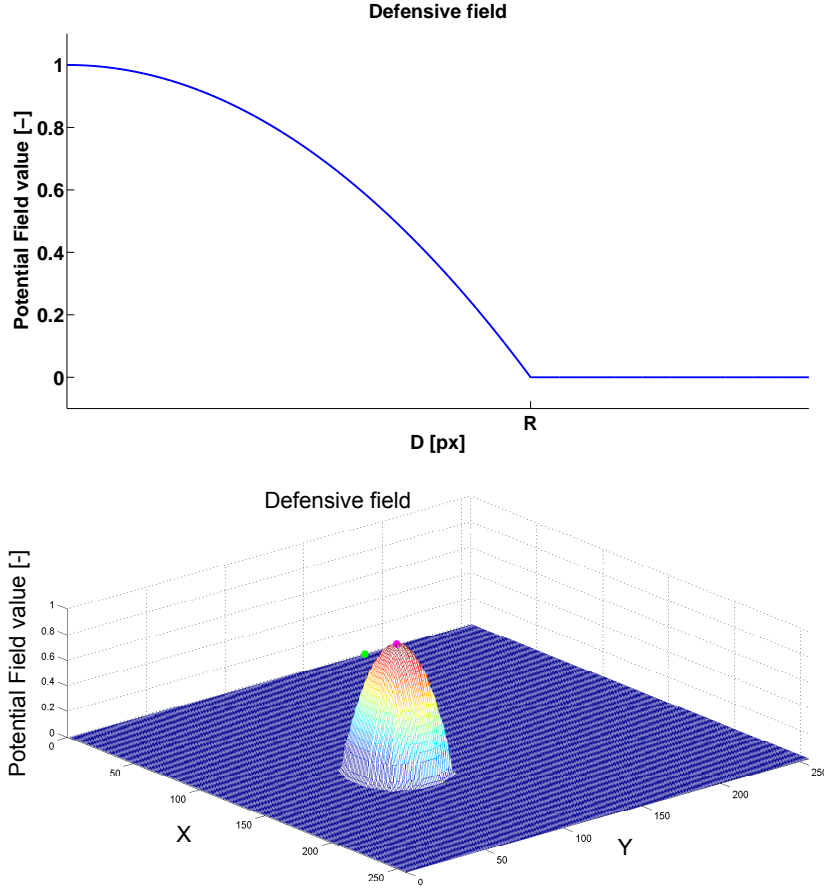


Figure 5.7.: The Potential Function  $PF_{defensive}$  generated by enemy unit given the distance  $D$

Evaluation function is calculated as the inverted Defensive field value:

$$P(Pos_{x,y} | PF_{defensive}) = 1 - PF_{defensive}(D). \quad (5.8)$$

### Collision field

All dynamic objects generate a repelling field for obstacle avoidance. Three radiuses are assigned to all dynamic objects. The size of the radius is based on the dimension of the object (unit dimension is provided through JNIBWAPI). The

value of the repelling field is determined by the distance between the agent and dynamic object. An illustration is shown in Figure 5.8.



Figure 5.8.: Illustration of collision radiuses

Radiuses are calculated as follows:

$$\mathbf{HardR}(Red) = 2 \cdot \max(UD_{left}, UD_{up}), \quad (5.9)$$

$$\mathbf{MediumR}(Orange) = 3 \cdot \max(UD_{left}, UD_{up}), \quad (5.10)$$

$$\mathbf{SoftR}(Green) = 5 \cdot \max(UD_{left}, UD_{up}), \quad (5.11)$$

where  $UD_{left}$  is an acronym for Unit Dimension, measured from the center to the edge of the unit (in our case it is the left and the upper edge).

The potential field  $PF_{collision}$  (depicted in Figure 5.9) at the distance  $D$  (in pixels) from the center of the object is calculated as:

$$PF_{collision} = \begin{cases} 0 & \text{if } D > \mathbf{SoftR} \\ 0.4 - 0.4 \frac{D - \mathbf{MediumR}}{\mathbf{SoftR} - \mathbf{MediumR}} & \text{if } D \in \langle \mathbf{SoftR}, \mathbf{MediumR} \rangle \\ 0.9 - 0.5 \frac{D - \mathbf{HardR}}{\mathbf{MediumR} - \mathbf{HardR}} & \text{if } D \in \langle \mathbf{MediumR}, \mathbf{HardR} \rangle \\ 1 - 0.1 \frac{D}{\mathbf{HardR}} & \text{if } D \leq \mathbf{HardR} \end{cases} \quad (5.12)$$

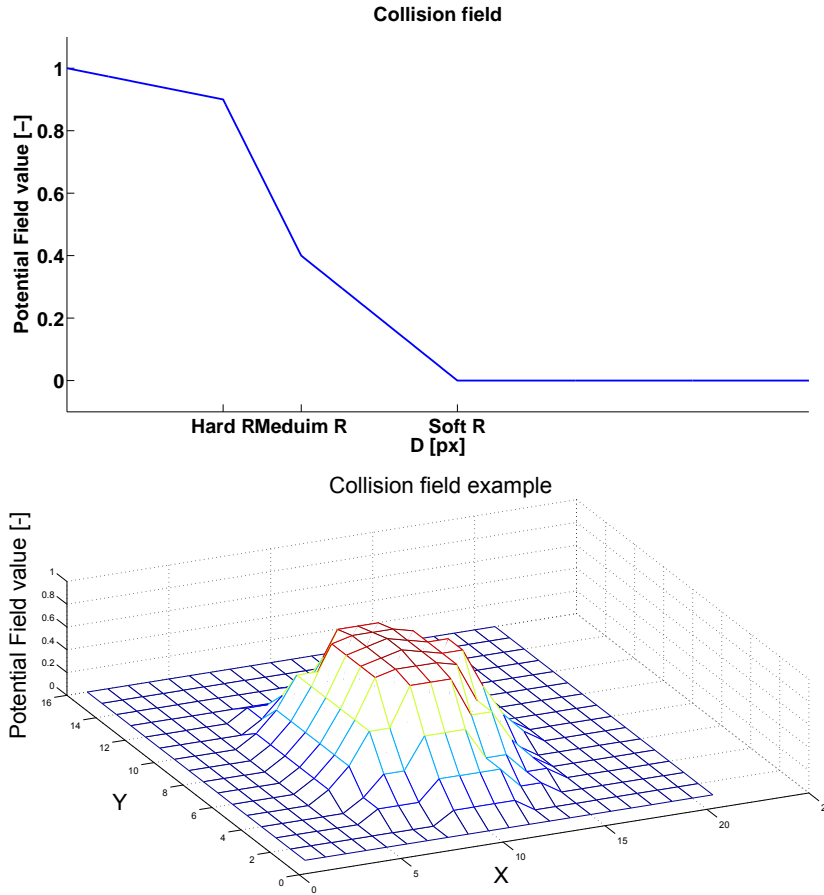


Figure 5.9.: The Potential Function  $PF_{collision}$  generated by enemy unit given the distance  $D$

Evaluation function is calculated as:

$$P(Pos_{x,y}|PF_{collision}) = 1 - PF_{collision}(D). \quad (5.13)$$

### Formation field

A group of agents generates an attractive field. This field addresses several issues at once. The field acts as a navigation, as a synchronizer of agent movements, and it also specifies the shape of the spatial deployment of the group. The construction of the field is based on the group average position, group size, and on the current group objective (objective position is assigned by a higher logic to the group). A construction of this field is illustrated in Figure 5.10. A link between average group position and objective position is created. For this line a perpendicular abscissa is constructed. The highest potential is precisely on this abscissa. Note, the position of the abscissa is shifted closer to the objective; hence, agents are driven forward to the objective position.



Figure 5.10.: Construction of formation abscissa example

The potential field  $PF_{collision}$  (depicted in Figure 5.11) at the distance  $D$  (in pixels) from the abscissa is calculated as:

$$PF_{formation} = \left(1 - \frac{D}{MD_{max}}\right)^4, \quad (5.14)$$

where  $MD_{max}$  is the maximal possible distance on the given map, which corresponds to the length of its size diagonal line. Notice, the distance  $D$  can be gathered in two ways.  $D$  is the distance from the abscissa, if a projection of the agent position lies on the abscissa. Otherwise  $D$  is the distance from an average group position. This condition guarantees that agents stay close to the average group position.



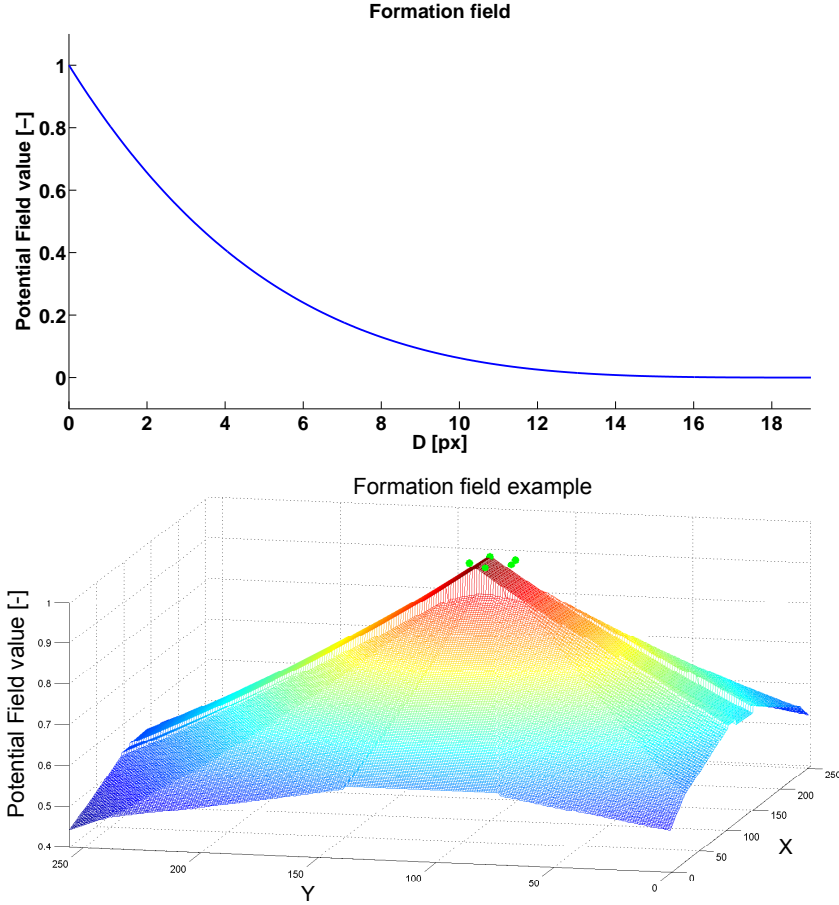


Figure 5.11.: The Potential Function  $PF_{formation}$  generated by enemy unit given the distance  $D$

Evaluation function is calculated as:

$$P(Pos_{x,y}|PF_{formation}) = PF_{formation}(D). \quad (5.15)$$

### Encirclement field

All of the enemy units generate a symmetric surrounding field. The highest potential is in the ring around the unit with the radius of the agent Maximum Sight Range (MSR). The Encirclement Potential Field (depicted in Figure 5.12) at the distance  $D$  (in pixels) from the center of the last known enemy position is calculated as:

$$PF_{encirclement} = \begin{cases} \left(1 - \frac{D}{MD_{max}}\right)^2 & \text{if } D \geq MSR \\ \left(\frac{D}{MD_{max}}\right)^4 & \text{if } D < MSR \end{cases} \quad (5.16)$$

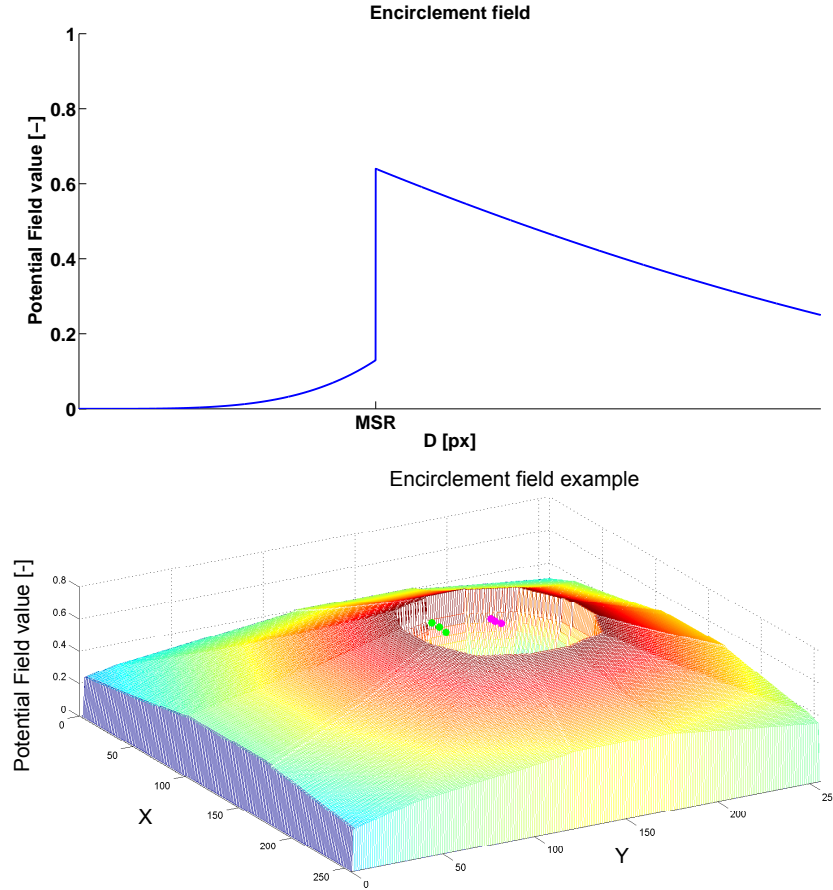


Figure 5.12.: The Potential Function  $PF_{encirclement}$  generated by enemy unit given the distance  $D$

Evaluation function is calculated as:

$$P(Pos_{x,y}|PF_{encirclement}) = PF_{encirclement}(D). \quad (5.17)$$

#### 5.1.4. Assigning granularity of time and space in the environment

When designing the agent, we had to decide what resolution we will use for the potential field and size of the area around the agent that will be checked. Training map used in this thesis has a size  $64 \times 64$  build tiles ( $2048 \times 2048$  pixels), which is the smallest map size in StarCraft. After initial evaluation I decided to use  $8 \times 8$  pixel for each tile in the potential field. This resolution is accurate enough for an agent to be able to move smoothly. In addition, as it is mentioned in Section 3.2.1, this resolution is also provided directly by JNIBWAPI.

Regarding the time granularity, an agent can check its own position plus additional area, which is determined by the size of the Area Offset (illustrated in Figure 5.13).

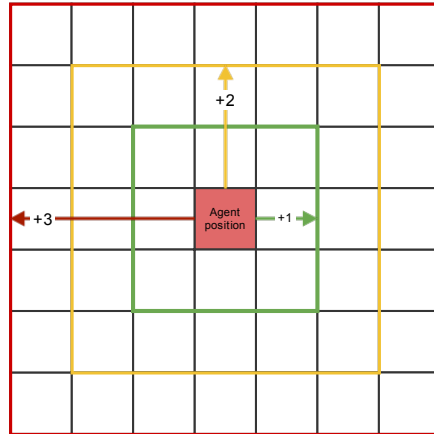


Figure 5.13.: Illustration of Area Offset

The minimal Area Offset would be +1. In this case, the agent checks only 8 nearest fields. The extension of the Area Offset results in a significant increase of computational time and memory usage. On the other hand, the use of +1 offset is somehow short-sighted. Thus, in our implementation, the Area Offset equals to +4 which has been found as a good trade-off. To reduce the computational time to compute the potential fields, evaluation are called after each fifth game frame.

### 5.1.5. The agent of the system

The agent system is based on Finite State Machine as described in Section 5.3. For the agent's state: Move or Flee, the agent actions are based on PFs. When deciding actions for agent, the potentials in surrounding tiles (given by the Area Offset as described above) are compared. The agent moves to the center of the tile with the highest potential, or it is idle in a case that the current tile is the highest. If an enemy unit is within a fire range, the agent enters the Fire state.

In the Fire state, the behavior of the agent is conditioned by unit cool-down on attack. When cool-down is zero, the agent chooses the enemy unit to attack and attack it. Otherwise agent behavior is chosen based on the type of the enemy. If the type is appropriate for using the Kiting strategy, then the agent state is switched to Flee state. Otherwise agent is idling.

The Kiting strategy is based on the movement of the units around to make the enemy chase them and thus not be able to attack as much, or not at all. This is often used by ranged units against melee units in a combination with the move-shoot reactive control [43].

Some of the units, like Lurkers, Siege Tanks etc., have a completely different control form and roles. The general agent can be extended according to the unit type, so that each type of unit could use their special abilities. In this thesis, only general agent has been described and implemented.

### 5.1.6. The architecture of the Multi-Agent System

The overall high level tactic and strategy is beyond scope of this thesis. However, there is the Group Tactic module which performs some tasks belonging to this area of control. The implemented tasks are: Target state prediction, Focusing fire, Encirclement group of enemies, and Group Logic responsible for determining the types of the actions, that units in a group should perform (e.g., attack, retreat, surround the enemy etc.). The Group Tactic is explained in Section 5.4.

## 5.2. Limitation

One of the shortcomings of the PFs approach is the local optima problem, see Section 4.1. The proposed solution to the local optima problem is to generate a negative trail on a specified number of the previous visited tiles. The trail depreciates the evaluation of visited positions, and therefore, it pushes the agent away from the local optima. In this work, this trail is implemented with the following features. The 15 last positions are saved for each agent (positions are in walkability resolution). Then, instead of adding any fixed value to the overall PF by the trail, values in the Collision Potential Field are reduced by 20 percentage points on positions that are contained in the agent trail. Agent can react only on its own trail. An illustration of this trail is shown in Figure 5.14.



Figure 5.14.: Negative trail example

However, this solution does not eliminate the problem. If agent faces some potential field inconsistencies or noisy values, then this negative trail can push them trough. On the other hand, if there is a larger obstacle, agent may still stuck in a local optima. Typical example is illustrated in Figure 5.15.

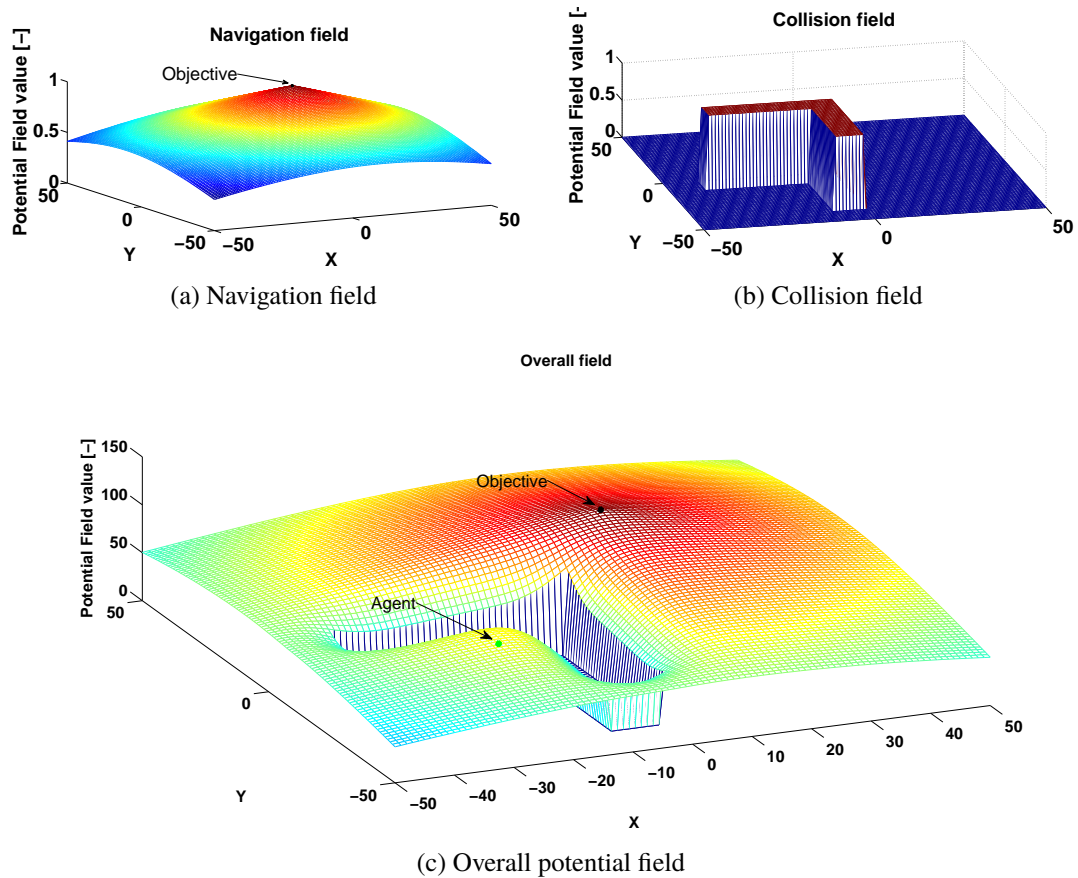


Figure 5.15.: An illustration of the local optimal problem

To fully eliminate this problem, some kind of path finding should be incorporated or completely abandon potential field concept and construct the agent based on a different approach.

### 5.3. Finite State Machine

To assign different agent behavior, I propose utilized the Finite State Machine (FSM) to extend Multi-Agent Potential Fields method. The structure of the FSM is shown in Figure 5.16.

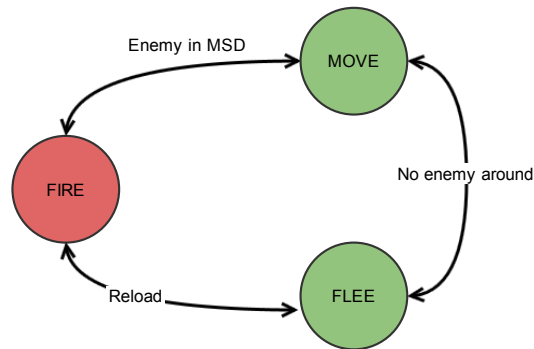


Figure 5.16.: State diagram for a agent

FSM states and state transitions are as follows:

- **Move state:** This is the default state. The agent uses the aforementioned described Potential fields to avoid collisions, reach the goal or seek enemy units. Firstly, the agent checks the bounded area in its MAPF (calculated as an overall Potential field (4.1) from the individual fields described in Section 5.1.2) as it is described in Section 5.1.4. Finally, the agent moves to the center of the tile with the highest potential, or it is idling if the current tile is the highest one.
- **Flee state:** This state is structurally the same as the Move state. The difference is in the overall PF construction. It is constructed from the defensive PF instead of the aggressive one (explained in detail in Section 5.1.3) In addition, the Formation field is not taken into account. The agent in this state is trying to avoid enemies as much as possible.
- **Fire state:** The agent in this state attacks an enemy its range. To increase efficiency, the attack coordination and damage prediction are used. Attacks are focused on enemies with a low Hit Points. Additionally no-overkill (providing more damage than necessary) mechanism is implemented.
- **Move→Fire:** If there is an enemy unit within the circle with the radius Maximum Shooting Distance (MSD) and centered on the agent.
- **Move→Flee:** If the agent is assigned the role of scout.
- **Fire→ Move:** If there is no enemy unit within the circle with the radius MSD and centered at the agent.
- **Fire→Flee:** If the agent has a non-zero attack cool-down and the agent unit MSD is noticeably greater that the enemy unit  $eMSD$  ( $eMSD < 0.8MSD$ ).
- **Flee→ Move:** If there is no enemy unit within the circle with the radius MSD and centered at the agent.
- **Flee→Fire:** If the agent has a zero attack cool-down and there is an enemy unit within the circle with the radius MSD and centered at the agent.

## 5.4. Group Tactic

Assigning unit to a group determines the formation of that group, common action (like attack, encirclement or retreat) and others. Since there is no high strategy level that can assign unit to the groups, units are assigned to one group.

### 5.4.1. Target state prediction

The need to save information about enemy units is given by the effort to give the agent addition abilities and maximize the efficiency of the focused fire. As it was mentioned in the description of the Encirclement field in Section 5.1.3, the purpose is to encircle an enemy group and gain favorable position for attack. The situation is complicated by the fact that the radius of the generated field is higher than the line of sight, and therefore, our agents can not see enemy units. Here, the target state prediction takes place. The last seen position of the enemy is saved and a group operates with the assumption that this position is valid. Based on this position, the group can try to encircle the enemies.

Another type of stored information is health status. The agents in the Fire state, which picked an enemy unit to attack, updates this enemy health status. The update is done as a damage prediction:

$$HP_{predicted} = HP_{current} - (E_{armor} - A_{damage}), \quad (5.18)$$

where  $HP$  is the Hit Points indicator at the given time,  $E_{armor}$  is the armor value of specific unit type + bonus armor given from opponent upgrades,  $A_{damage}$  is the damage amount + bonus damage given from the AI upgrades. It is worth mentioning that the damage amount is not constant but is chosen randomly between the minimum and maximum damage. The exact formula for damage calculation is  $damage = min_{dmg} + p(max_{dmg} - min_{dmg})$ , where  $p \in \langle 0, 1 \rangle$ . Notice, the  $HP_{current}$  in equation (5.18) is the actual unit Hit Points status or the previously predicted Hit Point status (if some agent has already picked unit as its target). The motivation behind this is that the agents do not waste attack on enemy units that are going to be killed in the next frame (another agents will attack them).

### 5.4.2. Focusing fire

The enemy units can be marked as focused. When the agent is in the state Fire and it is choosing the target to attack, the focused unit has a higher priority and it is chosen as the next target for the agent. The agent also looks for a focused unit in its neighborhood (150% of Maximum Shooting Distance of the unit). When the agent finds focused enemy unit and the agent is not under attack, it tries to attack the focused unit. The amount of the unit damage in StarCraft is not depended on health of the unit, and therefore, nearly dead enemy deals the same amount

of damage as healthy one. Focusing the fire is vital to eliminate enemy units as quickly as possible.

### 5.4.3. Encirclement and Group logic

To gain advantage before entering fight, a group can decide to encircle an enemy group. The encirclement gives a better position for agents and partly prevents escape of the enemy units. On the other hand, encirclement is not guaranteed and can lead to the very opposite situation, where agents are spread and forced to fight individually. To evaluate the situation and decide, proper response, the Group logic is used. Decision-making of the group is based on the Fuzzy logic. The inputs of the group controller are: Combat outcomes predictor output, Placement function output and Damage taken by the group. Fuzzyfication (finding the degree of membership of the input in predefined fuzzy sets) of these inputs is described below.

#### Combat outcomes predictor output

Evaluation of the enemy threat and prediction of the combat outcome is a substantial part of decision-making process. There are various combat simulators and predictors like [44] where possibilities of each unit are taken into account and a result is based on the state space search. However, in this thesis, the straight-forward evaluation function is used as a simplified estimation. Using a unit hit points, attack values and cool-down periods, Kovalsky and Bruno [19] proposed the following evaluation function for the combat games. Based on the life-time damage a unit can inflict:

$$LTD = \sum_{u \in U_A} Hp(u) \cdot \frac{Dmg(u)}{Cldwn(u)} - \sum_{u \in U_E} Hp(u) \cdot \frac{Dmg(u)}{Cldwn(u)}, \quad (5.19)$$

where  $U_A$  and  $U_B$  are the units controlled by the player  $A$  and player  $B$ ,  $Hp(u)$  are the Hit Points of the unit  $u$ ,  $Dmg(u)$  is the damage of the unit  $u$ , and  $Cldwn(u)$  is the maximal cool-down of the unit  $u$  attack rate.

Results of the estimation  $LTD$  is used as the input for the fuzzyfication. The membership functions map input variables to the degree of the membership, in the fuzzy set, between 0 and 1. The membership functions related to the combat outcome prediction are:

$$f_{Enemy\ superiority} = \begin{cases} 1 & LTD \leq -200 \\ \frac{LTD}{-200} & -200 < LTD < 0 \\ 0 & LTD \geq 0 \end{cases}, \quad (5.20)$$



$$f_{Balance} = \begin{cases} 0 & LTD \leq -200 \\ \frac{LTD+200}{200} & -200 < LTD < 0 \\ 1 - \frac{LTD}{200} & 0 < LTD < 200 \\ 0 & LTD \geq 200 \end{cases}, \quad (5.21)$$

$$f_{Allies\ superiority} = \begin{cases} 0 & LTD \leq 0 \\ \frac{LTD}{200} & 0 < LTD < 200 \\ 1 & LTD \geq 200 \end{cases}. \quad (5.22)$$

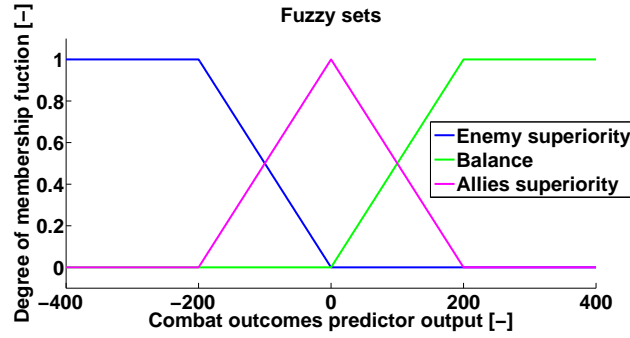


Figure 5.17.: Illustration of the membership functions

### Placement function output

As an indicator of the proper unit placement, the Encirclement potential field value is used. Agents in the group save the value of the Encirclement potential field on its current position. The values of the current placement are divided by the maximal possible value to determine whether units are deployed optimally.

$$PR = \frac{\sum_{u \in U_A} P(Pos_{x,y}(u) | PF_{encirclement})}{N \cdot \max(PF_{encirclement})}, \quad (5.23)$$

where  $N$  is number of agents in group.

The membership functions related to unit placement is:

$$f_{Placement} = \begin{cases} 0 & PR \leq 0 \\ PR & 0 < PR < 1 \\ 1 & PR \geq 1 \end{cases}. \quad (5.24)$$

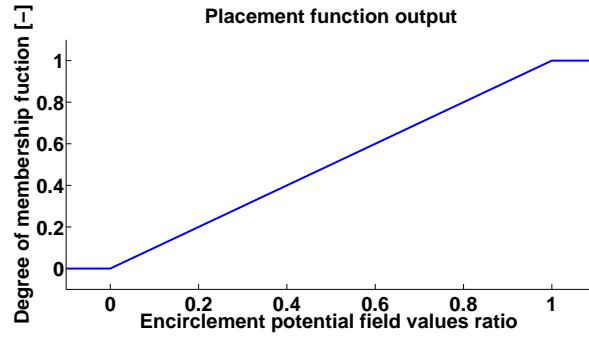


Figure 5.18.: Illustration of membership function

### Damage taken by the group

The encirclement operation is risky and the enemy may engage our agents before they get into the desired position. The damage of membership function taken by the group through time is an indicator that is utilized to decision making. The taken damage is measured as a percentage of the total Hit Points of the group. The membership function related to taken damage is:

$$f_{Damage} = \begin{cases} 0 & PR \leq 0\% \\ 1 - \frac{\sum_{u \in U_A} Hp(u)}{\sum_{u \in U_A} Hp_B(u)} & 0\% < PR < 20\% \\ 1 & PR \geq 20\% \end{cases}, \quad (5.25)$$

where  $Hp_B(u)$  are the Hit Points of the unit  $u$  before encounter.

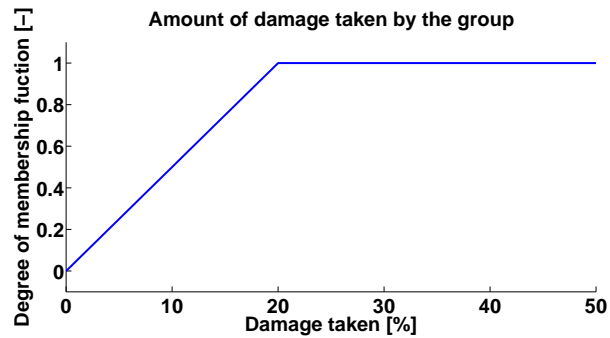


Figure 5.19.: Illustration of membership functions

### Decision making

The output actions can be attack, retreat or wait. The degree of the membership for each output action is evaluated by these rules:

$$Attak = (f_{Allies\ superiority} AND f_{Placement}) OR f_{Damage}, \quad (5.26)$$

$$Retreat = f_{Enemy\ superiority} AND f_{Damage}, \quad (5.27)$$

$$Wait = NOT f_{Damage}. \quad (5.28)$$

The highest degree of this rules determines the next action of the group.

## 5.5. Structure of the implemented AI overview

The Overview of the structure of the Java project, where the proposed AI is implemented, is described as follows:

**AIClient** (120 LC) - Client used to create connection to BWAPI and StarCraft,

- **JNIBWAPI** - Java wrapper of the BWAPI,
- **OverMind** (80 LC) - The main method of the AI, which contains all data about the game and all managers (currently only Unit Manager is functional),
  - **UnitManager** (440 LC) - Unit Manager controls all units in the game,
    - \* **ScoutGroup** (1000 LC) - group of units focused on explotarion and reconnaissance,
    - \* **Group** (750 LC) - group of the units focused on combat,
      - **Agent** (900 LC) - MAPF based agent,
      - **Target** (80 LC) - Enemy unit data are saved in this object,

where LC corresponds to app. number of code lines in specific class. For more detailed view see the CD provided with the thesis.

## Chapter 6.

# 6 Experiments and Results

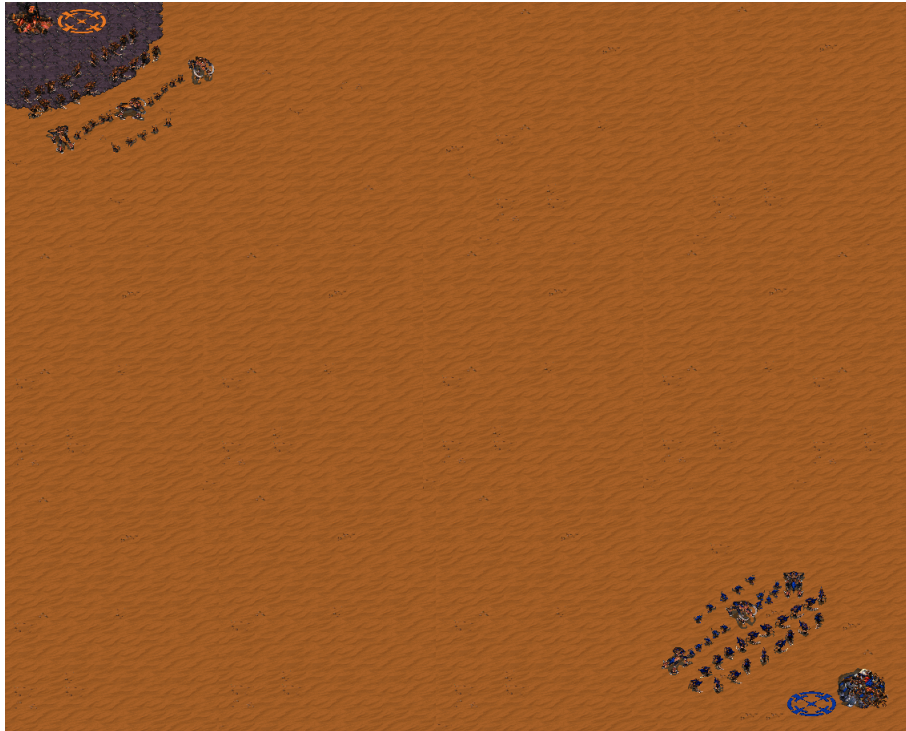
---

Explanation, description and results of all main experiments conducted in this thesis are presented in this chapter.

### 6.1. Experiment overview

In order to evaluate quality and efficiency of the designed AI, I conducted several experiments. The experiments are designed as training maps. All of the training maps are created with the StarCraft Campaign Editor, which is included in an installation of the StarCraft. The experiments are set on maps  $64 \times 64$  build tile size, with 1v1 player settings. To prevent unbalanced experiments, the maps are used with identical strength of the players; hence, the number of units of the same type on both side are equal. As it was mentioned earlier in Section 2.2, the kind of the race in StarCraft has a huge impact on the gameplay style, and therefore, there are different requirements of the Reactive Control for different races. For these experiments, I decided to use the units from the Zerg race. The reason is that Zerg unit types contain large variety of ranged and melee units. The agent should manage to control these two basic types, whose control is significantly different.

A basic training map decomposition is illustrated in Figure 6.1. Both starting locations are placed into opposite corners and facing each other. The identical unit groups are placed on those locations. An objective for these groups is to reach the enemy base (on start location) and destroy it. Groups move towards their objective and inevitably encountered with the group of the opponent.



(a)



(b)

Figure 6.1.: Example of the training maps, (6.1a) illustrates the general test training map. Figure (6.1b) illustrates modified test map

Notice, the training map does not contain any complicated terrain or any other obstacles that could prevent correct unit navigation.

Descriptions of all units used in the training maps are presented here. The types of the units used on training maps are shown in Figure 6.2.



(a) Hydralisk (b) Zergling (c) Ultralisk

Figure 6.2.: Military units used on training maps

Table 6.1 shows some information about the unit characteristics used in the training maps. The Range is the Maxim Shooting Distance (MSD) and the Cooldown is the weapon cool down. The attack value is the maximum amount of the damage that a unit weapon can deal per attack. Values such as the Range and Sight are quantified by a walkable distance in the number of tiles. The Cooldown is quantified by the number of the game frames.

Table 6.1.: Overview of units properties

	Type	Hit Points	Armor	Attack	Range	Cooldown	Sight
Hydralisk	Medium Ground Unit	80	0	10	4	15	6
Zergling	Small Ground Unit	35	0	5	1	8	5
Ultralisk	Large Ground Unit	400	1	20	10	15	7

Some attributes may differ, if units have got upgrades. In this thesis, there are no upgrades used on the training maps. The values of the cooldown listed here are the base values, as StarCraft randomizes the cooldown. It takes the base cooldown and adds a value between (-1) and (+2)<sup>1</sup>

To evaluate various aspects of the proposed AI strategy we need to inspect several key attributes of the test scenario. The general test scenario presented above may vary in these attributes:

- Unit types used in the scenario (homogenous group vs. mixed types, ranged units vs. melee units, etc.)
- Size of the group (the bigger the group is, the more unwieldy it becomes).
- Group tactics (there can be three basic scenarios of the fight: battle an open field, where both groups attack each other, attack on defensive formation or defensive role of the group).

<sup>1</sup><https://code.google.com/p/bwapi/wiki/StarcraftGuide>

- Opponent AI (Original AI included in StarCraft or other bots).

Regarding these criteria, experiments are divided as follows. With respect to the type of units, experiments are divided to a range group (composed of Hydralisks), melee group (composed of Ultralisks and Zerglings) and mixture of all units types. According to the size of the groups, experiments are divided to small scale combat (5 vs. 5 unit), medium scale combat (20 vs. 20) and large scale combat (50 vs. 50).

After some initial tests, I decided to reduce group tactics into two parts: 1) attack on contra attacking enemy; 2) and attack on defensive enemy (encirclement tactics described in Section 5.4.3). The main reason for this is that the AI proposed in this thesis does not support a defensive behavior. Encounter with enemies leads to a counter attack and the situation is then basically the same as in the attacking scenario. Additionally, encirclement tactic makes no sense when a group of units is too small. Therefore, experiments with this tactic are conducted only for the medium and large scale combat scenarios. This setup gives us experiments for each opponent AI. the structure of this setup is illustrated in Table 6.2. Each cell in the table represents a scenario for which a series of tests has been conducted.

Table 6.2.: Structure of the experiment

		Size		
		Small	Medium	Large
Type	Melee	5 vs. 5 (5 Zerglings per group)	20 vs. 20 (18 Zerglings + 2 Ultralisks per group)	50 vs. 50 (40 Zerglings + 10 Ultralisks per group)
	Range	5 vs. 5 (5 Hydralisks per group)	20 vs. 20 (20 Hydralisks per group)	50 vs. 50 (50 Hydralisks per group)
	Mix	5 vs. 5 (3 Zerglings + 2 Hydralisks per group)	20 vs. 20 (10 Zerglings + 2 Ultralisks + 8 Hydralisks per group)	50 vs. 50 (30 Zerglings + 5 Ultralisks + 15 Hydralisks per group)

The series of the test for each scenario consists of 50 trials. The following data are record as the results of particular evaluation trial: the mission status (win or lose), the number of the enemy units left, and the number of own units left.

## 6.2. Conducted experiments

The opponents considered in the performed experiments are the Original built-in AI (OAI), TSCmoo AI (TSCAI) and OpprimoBot AI (OBAI). The opponent selec-

tion was difficult from the following reasons. There is a few bots focused on Zerg race. In addition, situation is complicated by the fact that these bots are designed to play full game with normal starting conditions. In our testing scenarios, there are groups of units locked in small area, hence combat is forced. Other aspects of the game are missing. This circumstance may lead to unexpected behavior of the bot. Therefore, only opponents that are able to behave normally are chosen. Note, that to be able perform combat scenario test, the perfect visibility is enabled (provided by the BWAPI) for both AIs. The opponents are described below. Additionally, the test scenarios for non OAI are reduced to the Small and Medium scale combat scenarios and due to the fact that I have no control over opponents AI, the tactic for the group is set to the direct attack. The reason for the combat scale reduction is that an agent proposed in this thesis is not able to work properly in a large formation. Hence, it results to a poor performance in the Large combat scenarios against OAI. More detailed explanation is discussed in 6.4.

### **OpprimoBot**

The OpprimoBot<sup>2</sup> (previously named BTHAI) is a bot that is based on the work of Dr. Johan Hagelbäck. The version of the bot that is used in this thesis is ver. 15.4 . The agents are based on the MAPFs approach. The bot was originally focused on the Terran race. In the latest version, all races are supported.

### **TSCmoo**

TSCmoo bot<sup>3</sup> is a participant in the Student StarCraft AI Tournament(SSCAIT) 2015 [45]. There is a little knowledge about this bot. It probably uses a case based reasoning, influence maps, and others techniques. In the SSCAIT 2015 in “Mixed division” this bot has the win ratio app. 77% (to the date 4.5.2015).

## **6.3. Results of the experiment**

In this section, a summary of the experiments is presented. For a more detailed view on experiments results see the CD provided with the thesis, where all measurements are recorded. Additionally, all replays from all conducted tests presented in this thesis are saved and stored on this CD. Abbreviations used in these tables are described as follows: AUL corresponds to the number of allied units left on the end of the test and EUL corresponds to the number of enemy units left in the end of the test (in the both cases, the numbers are presented in a form of the mean and standard deviation.).

---

<sup>2</sup><http://aiguy.org/OpprimoBot.html>

<sup>3</sup><https://github.com/tscmoo>



Table 6.3.: Summary of the experiments for opponent OAI

	win rate [%]	AUL [-]	EUL [-]
Range	96	$2.16 \pm 0.841$	$0.08 \pm 0.444$
Melee	28	$0.52 \pm 0.973$	$1.26 \pm 0.964$
Mix	90	$1.68 \pm 0.74$	$0.1 \pm 0.303$
Overall	71.3	$1.453 \pm 1,096$	$0.48 \pm 0.841$

(a) Summary of the experiment results for Small scale combat scenario (Attack tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	40	$1.56 \pm 2.242$	$2.3 \pm 2.296$
Melee	44	$0.88 \pm 1.624$	$2.04 \pm 2.407$
Mix	82	$4.7 \pm 2.771$	$0.38 \pm 0.878$
Overall	55.33	$2.38 \pm 2.799$	$1.573 \pm 2.149$

(b) Summary of the experiment results for Medium scale combat scenario (Attack tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	88	$6.6 \pm 3.344$	$0.48 \pm 1.445$
Melee	26	$0.96 \pm 1.958$	$2.12 \pm 2.352$
Mix	72	$2.82 \pm 2.43$	$0.98 \pm 1.789$
Overall	62	$3.46 \pm 3.524$	$1.1933 \pm 2.01$

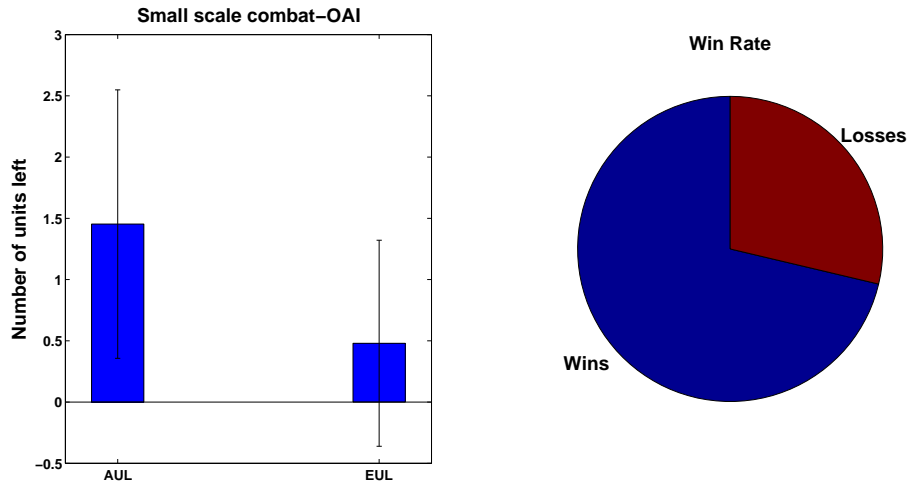
(c) Summary of the experiment results for Medium scale combat scenario (Encirclement tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	0	$0.02 \pm 0.143$	$13.4 \pm 3.642$
Melee	92	$3.5 \pm 2.196$	$0.18 \pm 0.628$
Mix	24	$1.74 \pm 3.567$	$6.3 \pm 4.726$
Overall	38.66	$1.753 \pm 2.794$	$6.626 \pm 6.419$

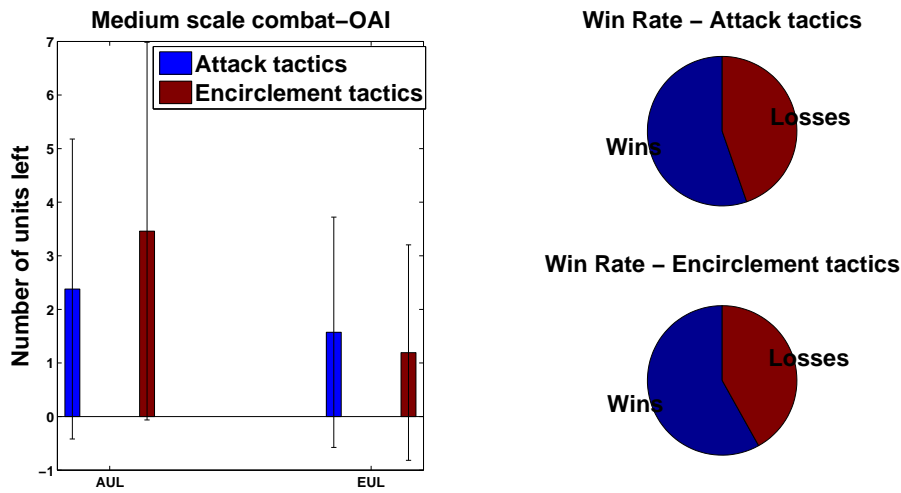
(d) Summary of the experiment results for Large scale combat scenario (Attack tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	20	$1.7 \pm 3.699$	$5.38 \pm 3.978$
Melee	82	$5.66 \pm 3.863$	$0.74 \pm 1.816$
Mix	28	$1.72 \pm 3.037$	$4.02 \pm 3.809$
Overall	43.33	$3.026 \pm 3.991$	$3.38 \pm 3.857$

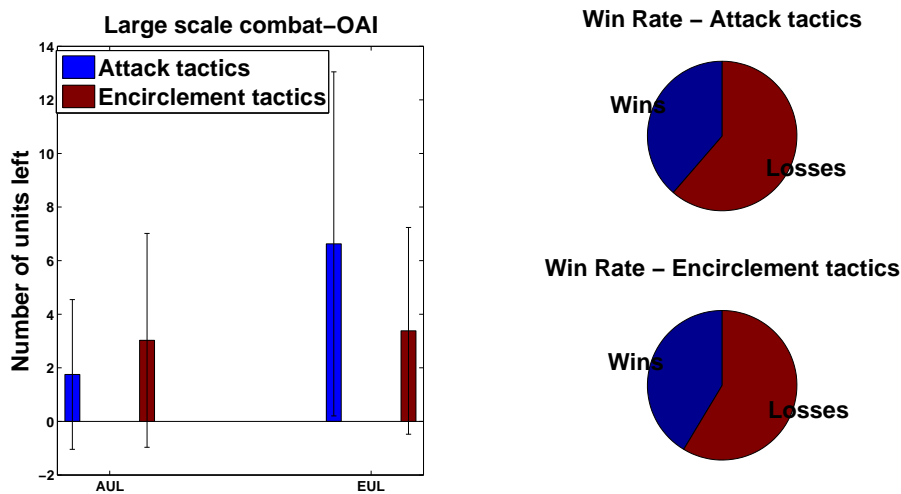
(e) Summary of the experiment results for Large scale combat scenario (Encirclement tactics)



(a) Visualization of the experiment results for Small scale combat scenario (Attack tactics)



(b) Visualization of the experiment results for Medium scale combat scenario (Attack tactics)



(c) Visualization of the experiment results for Large scale combat scenario (Attack tactics)

Figure 6.3.: Visualization of the experiment results for opponent OAI

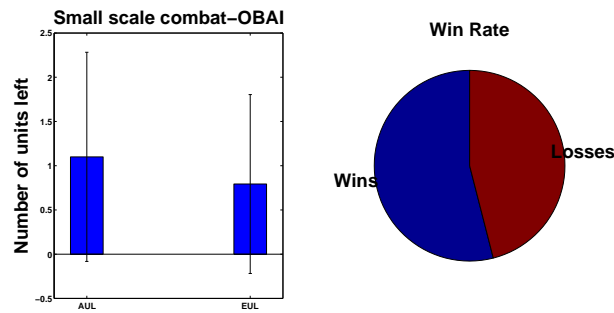
Table 6.4.: Summary of the experiments for opponent OBAI

	win rate [%]	AUL [-]	EUL [-]
Range	50	$0.82 \pm 0.962$	$0.72 \pm 0.833$
Melee	30	$0.62 \pm 1.054$	$1.36 \pm 1.173$
Mix	82	$1.88 \pm 1.136$	$0.3 \pm 0.677$
Overall	54	$1.10 \pm 1.182$	$0.793 \pm 1.011$

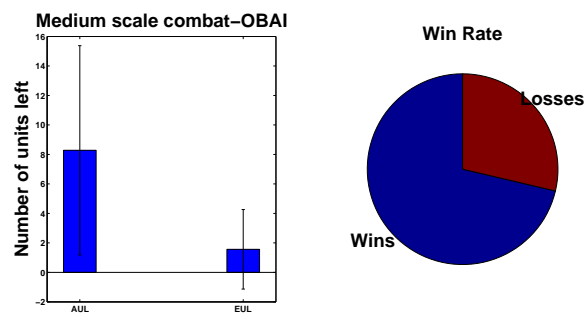
(a) Summary of the experiment results for Small scale combat scenario (Attack tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	34	$1.96 \pm 3.504$	$3.64 \pm 3.173$
Melee	100	$15.22 \pm 4.652$	$0 \pm 0$
Mix	80	$7.66 \pm 5.374$	$1.06 \pm 2.235$
Overall	71.333	$8.28 \pm 7.095$	$1.566 \pm 2.697$

(b) Summary of the experiment results for Medium scale combat scenario (Attack tactics)



(a) Visualization of the experiment results for Small scale combat scenario (Attack tactics)



(b) Visualization of the experiment results for Medium scale combat scenario (Attack tactics)

Figure 6.4.: Visualization of the experiment results for opponent OBAI

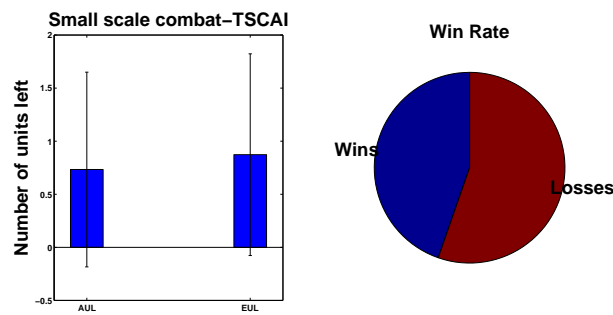
Table 6.5.: Summary of the experiments for opponent TSCAI

	win rate [%]	AUL [-]	EUL [-]
Range	74	$1.38 \pm 0.923$	$0.46 \pm 0.862$
Melee	28	$0.46 \pm 0.862$	$1.32 \pm 1.077$
Mix	32	$0.36 \pm 0.562$	$0.84 \pm 0.68$
Overall	44.666	$0.733 \pm 0.917$	$0.873 \pm 0.95$

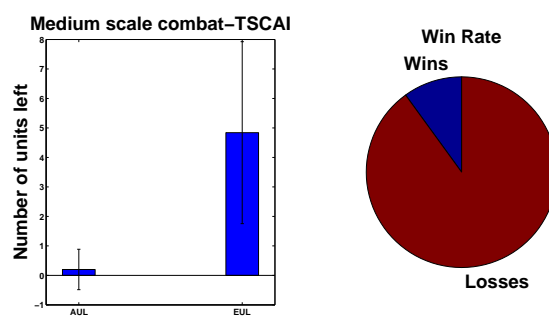
(a) Summary of the experiment results for Small scale combat scenario (Attack tactics)

	win rate [%]	AUL [-]	EUL [-]
Range	16	$0.42 \pm 1.031$	$3.94 \pm 2.385$
Melee	2	$0.02 \pm 0.141$	$6.92 \pm 3.433$
Mix	12	$0.16 \pm 0.509$	$3.66 \pm 2.19$
Overall	10	$0.2 \pm 0.685$	$4.84 \pm 3.085$

(b) Summary of the experiment results for Medium scale combat scenario (Attack tactics)



(a) Visualization of the experiment results for Small scale combat scenario (Attack tactics)



(b) Visualization of the experiment results for Medium scale combat scenario (Attack tactics)

Figure 6.5.: Visualization of the experiment results for opponent TSCAI

The overall win rate (all scenarios combined) against OAI is 54.12%, against OBAI it is 62.66% and against TSCAI it is 27.33%.

## 6.4. Discussion

Results from the experiments against OAI are considered as the base performance indicator. The results indicate, that there is a gradual decline of the performance as the number of units in a group grows. This effect is partially caused by a local optima problem of the MAPF approach. An agent has often blocked path by another unit from its group, the agent is thus forced to look for alternative paths to reach its objective and avoid collision. This leads to a time consuming procedure since each agent moves in order to find its place in the group formation. With the increasing number of agents in the group, the collision probability grows and agents are often stuck in the local optima points on the used PF. When this situation takes place in a combat encounter, the agent can not find its way to the enemy, and therefore, it does not participate in the combat. Hence, the group size is effectively lower than it has been originally created. This affects particularly melee units types. A proper unit placement like encirclement can significantly improve the performance as it is shown in the results. However, this tactic is not appropriate in all situations and additional higher logic is required. In this thesis, the presented logic can evaluate failed attempt for the encirclement and retreat or attack. Unfortunately, in the test scenarios, where players forces are equivalent, it is usually too late and unsuccessful attempt leads to defeat in the scenario.

Experiments against TSCAI show the lack of adaptability of the formation approach. Agents are focused on nearest enemy unit and move toward it as a whole group in the formation. This mechanic can be exploited. TSCAI split formation into main formation and bait units. The bait units lure the group to attack them and the main group attacks from a more preferable position (e.g., on flank) a moment later. Since the level of the control proposed in this thesis is focused on the Reactive Control, groups have their objectives assigned manually. Hence, some kind of threat evaluation and objective adjustment should be present in the higher logic.

An important part of the MAPF solution of the agent are the weights. As it was mentioned before, the weights tuning is performed manually. Manual tuning takes a lot of time and there is not guarantee that the weights are tuned optimally (all the used settings of PFs including weights are stored on CD attached to this thesis). Incorrect weights could be a reason that the PF did not perform optimally.

# 7 Chapter 7.

---

## Conclusion

In this thesis, we proposed a complex Reactive Control AI based on a Multi-Agent Potential Field approach for the RTS game StarCraft. The goal was to design method of multi-agent movement control and coordination of groups of mobile units. The solution is based on existing concepts of the Potential fields that has been extended to Finite State Machine, Fuzzy logic based decision-making, and Bayesian function based evaluation function. This techniques are subsumed into the proposed solution to adjust behavior of the the multi-agent system. Multi-Agent Potential Fields system presented in Section 5.1 is capable to control the unit agents in a group to navigate around in simple environments, to avoid collision with static and dynamic objects, to create group formations, and to successfully fight the enemy units.

In the end, the experiments yielded rather disappointing results in terms of practical performance. The average win rate against the OAI is is 54.12%, , against the OBAI it is 62.66% and against the TSCAI it is 27.33%. On the other hand, the use of the proposed tactics leads to improvement of performance, namely it rises from 55.33% to 62% win rate in the Medium scale scenarios and the increase from 38.66% to 43.33% win rate in the Medium scale scenarios.

There were several limitation observed. The question is whether this problem lies in the design (the fundamental limitations of the PF approach e.g., local optima problem) or the implementation of the Multi-Agent Potential Fields (e.g., incorrect weights of PFs). After several revisions of the weights, the test results did not show any significantly improved performances. This empower the suspicion that the fault does indeed lie in the fundamental implementation of the Potential Fields.

However, the work that has been done in this thesis provides a good basis for future work for designing MAPF based AI for full scale RTS games.

## 7.1. Further work

Tuning weights manually turned out to be an endless task. Since there is no clear way how to evaluate performance difference as a reaction to weights change. For further work it would be useful to extend current MAPF system by some auto-tuning procedure like evolutionary algorithm.

It would also be useful to add pathfinding for agents, clusters of agents or the group depending on the computation time demand. Extension of the proposed solution to the Potential Flow approach would also be an interesting possibility. This would allow our multi-agent system to keep its structure and completely eliminate local optima problem.

StarCraft has been proven to be an excellent test platform, but the methods are applicable for almost all of RTS game. It would be interesting to see application of this approach attempted in different RTS games.

# Bibliography

- [1] H. Safadi, “Local Path Planning Using Virtual Potential Field,” Apr. 2007. [Online]. Available: <http://www.cs.mcgill.ca/~hsafad/robotics/>
- [2] “Fuzzy logic,” Apr. 2015, page Version ID: 655200038. [Online]. Available: [http://en.wikipedia.org/w/index.php?title=Fuzzy\\_logic&oldid=655200038](http://en.wikipedia.org/w/index.php?title=Fuzzy_logic&oldid=655200038)
- [3] “*Dune II*,” Feb. 2015, page Version ID: 645892103. [Online]. Available: [http://en.wikipedia.org/w/index.php?title=Dune\\_II&oldid=645892103](http://en.wikipedia.org/w/index.php?title=Dune_II&oldid=645892103)
- [4] “Command & Conquer,” Jan. 2015, page Version ID: 12190757. [Online]. Available: [http://cs.wikipedia.org/w/index.php?title=Command\\_%26\\_Conquer&oldid=12190757](http://cs.wikipedia.org/w/index.php?title=Command_%26_Conquer&oldid=12190757)
- [5] “Warcraft,” Mar. 2015, page Version ID: 12318926. [Online]. Available: <http://cs.wikipedia.org/w/index.php?title=Warcraft&oldid=12318926>
- [6] “StarCraft: Brood War wiki,” Feb. 2015, page Version ID: 645433201. [Online]. Available: [http://en.wikipedia.org/w/index.php?title=StarCraft:\\_Brood\\_War&oldid=645433201](http://en.wikipedia.org/w/index.php?title=StarCraft:_Brood_War&oldid=645433201)
- [7] D. Adams, “The State of the RTS,” *IGN PC. com. DisponÃvel em*, 2006. [Online]. Available: <http://www.ign.com/articles/2006/04/08/the-state-of-the-rts>
- [8] “Can Blizzard top itself with ‘StarCraft II?’ - Technology & science - Games - On the Level | NBC News.” [Online]. Available: <http://www.nbcnews.com/id/18925251/#.VPs5OOGS9m8>
- [9] “bwapi/bwapi.” [Online]. Available: <https://github.com/bwapi/bwapi>
- [10] J. Hagelback, “Multi-agent potential field based architectures for real-time strategy game bots,” *Blekinge Tekniska hÃ¶gskola*, 2012.
- [11] G. Synnaeve and P. Bessiere, “A Bayesian model for RTS units control applied to StarCraft,” in *2011 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug. 2011, pp. 190–196.
- [12] I. Gonzalez and L. Garrido, “Spatial Distribution through Swarm Behavior on a Military Group in the Starcraft Video Game,” in *2011 10th Mexican International Conference on Artificial Intelligence (MICAI)*, Nov. 2011, pp. 77–82.



- 
- [13] S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 4, pp. 293–311, Dec. 2013.
- [14] M. Buro, "Real-time strategy games: A new AI research challenge," in *International Joint Conferences on Artificial Intelligence*, 2003, pp. 1534–1535.
- [15] L. Perkins, "Terrain Analysis in Real-Time Strategy Games: An Integrated Approach to Choke Point Detection and Region Decomposition." *AIIDE*, vol. 10, pp. 168–173, 2010.
- [16] M. Sharma, M. P. Holmes, J. C. Santamaría, A. Irani, C. L. Isbell Jr, and A. Ram, "Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL." in *IJCAI*, vol. 7, 2007, pp. 1041–1046.
- [17] P. Cadena and L. Garrido, "Fuzzy case-based reasoning for managing strategic and tactical reasoning in starcraft," in *Advances in Artificial Intelligence*. Springer, 2011, pp. 113–124.
- [18] C. Miles and S. J. Louis, "Co-evolving real-time strategy game playing influence map trees with genetic algorithms," in *Proceedings of the International Congress on Evolutionary Computation, Portland, Oregon*, 2006, pp. 0–999.
- [19] D. Churchill, A. Saffidine, and M. Buro, "Fast Heuristic Search for RTS Game Combat Scenarios." in *AIIDE*, 2012.
- [20] W. Hu, Q. Zhang, and Y. Mao, "Component-based hierarchical state machine #x2014; A reusable and flexible game AI technology," in *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*, vol. 2, Aug. 2011, pp. 319–324.
- [21] D. Isla, "Handling complexity in the Halo 2 AI," in *Game Developers Conference*, vol. 12, 2005.
- [22] W. Zhe, N. Q. Kien, T. Ruck, and R. Frank, "Using Monte-Carlo Planning for Micro-Management in StarCraft," in *Proc. of the 4th Annual Asian GAME-ON Conference on Simulation and AI in Computer Games (GAMEON ASIA)*, 2012, pp. 33–35.
- [23] M. Massari, G. Giardini, and F. Bernelli-Zazzera, "Autonomous navigation system for planetary exploration rover based on artificial potential fields," in *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference*, 2004.
- [24] F. W. Heckel, G. M. Youngblood, and D. H. Hale, "Influence points for tactical information in navigation meshes," in *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM, 2009, pp. 79–85.

- [25] H. Danielsiek, R. StÅEer, A. Thom, N. Beume, B. Naujoks, and M. Preuss, “Intelligent moving of groups in real-time strategy games,” in *Computational Intelligence and Games, 2008. CIG’08. IEEE Symposium On*. IEEE, 2008, pp. 71–78.
- [26] L. Liu and L. Li, “Regional cooperative multi-agent q-learning based on potential field,” in *Natural Computation, 2008. ICNC’08. Fourth International Conference on*, vol. 6. IEEE, 2008, pp. 535–539.
- [27] E. A. Rathe and J. B. Svendsen, “Micromanagement in Starcraft using potential fields tuned with a multi-objective genetic algorithm,” 2012.
- [28] T. Nguyen, K. Nguyen, and R. Thawonmas, “Potential flow for unit positioning during combat in StarCraft,” in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, 2013, pp. 10–11.
- [29] S. Wender and I. Watson, “Applying reinforcement learning to small scale combat in the real-time strategy game starcraft: broodwar,” in *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*. IEEE, 2012, pp. 402–408.
- [30] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT Press, 1998.
- [31] U. Jaidee and H. Munoz-Avila, “Classq-l: A q-learning algorithm for adversarial real-time strategy games,” in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.
- [32] S. Wintermute, J. Xu, and J. E. Laird, “Sorts: A human-level approach to real-time strategy ai,” *Ann Arbor*, vol. 1001, no. 48, pp. 109–2121, 2007.
- [33] “Java Native Interface,” Apr. 2015, page Version ID: 656862187. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Java\\_Native\\_Interface&oldid=656862187](https://en.wikipedia.org/w/index.php?title=Java_Native_Interface&oldid=656862187)
- [34] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.
- [35] J. Hagelback, “Potential-field based navigation in starcraft,” in *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*. IEEE, 2012, pp. 388–393.
- [36] C. Thureau, C. Bauckhage, and G. Sagerer, “Learning human-like movement behavior for computer games,” in *Proc. Int. Conf. on the Simulation of Adaptive Behavior*, 2004, pp. 315–323.
- [37] M. Dorigo, *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*. Springer Science & Business Media, 2006, vol. 4150.
- [38] J. Hagelback and S. J. Johansson, “Using multi-agent potential fields in real-time strategy games,” in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 631–638.

- [39] R. Le Hy, A. Arrigoni, P. BessiÅšre, and O. Lebeltel, "Teaching bayesian behaviours to video game characters," *Robotics and Autonomous Systems*, vol. 47, no. 2, pp. 177–185, 2004.
- [40] D. M. Bourg and G. Seemann, *AI for game developers*. " O'Reilly Media, Inc.", 2004.
- [41] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: <http://dx.doi.org/10.1021/ac60214a047>
- [42] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "Smoothing by Example: Mesh Denoising by Averaging with Similarity-Based Weights," in *IEEE International Conference on Shape Modeling and Applications, 2006. SMI 2006*, Jun. 2006, pp. 9–9.
- [43] "Kiting." [Online]. Available: <http://wiki.teamliquid.net/starcraft2/Kiting>
- [44] M. Stanescu, S. P. Hernandez, G. Erickson, R. Greiner, and M. Buro, "Predicting Army Combat Outcomes in StarCraft." in *AIIDE*, 2013.
- [45] "[SSCAIT] Student StarCraft AI Tournament 2015." [Online]. Available: <http://www.sscaitournament.com/index.php?action=scores>

# A

## Appendix A.

---

# CD content

- DP\_Experiments - Contains test maps, results of the tests and replays of the all conducted tests.
- JNIBWAPI - Eclipse Java project. Contains AI implementation. The manual to this project in also included as the Manual.doc text file.
  - Methods\_of\_multi-agent\_movement\_control\_and\_coordination\_of\_groups\_of\_mobile\_units\_in\_a\_time-strategy\_games.pdf - Master thesis.