

A Multi-Objective Genetic Algorithm for Evaluating Build Order Effectiveness in Starcraft II

Jonas Schmitt

July 1, 2015

Outline

- 1 Motivation
- 2 Forward Simulation
- 3 Optimization
- 4 Results

Overview

1 Motivation

2 Forward Simulation

3 Optimization

4 Results

Starcraft II

- Military science-fiction real-time strategy game
- **Goal:** Producing the right combination of units (**Macromanagement**) to destroy the other player's units and structures in combat (**Micromanagement**)
- E-Sport scene with growing popularity (prize pools up to US\$170,000)
 - ⇒ Importance of **Balancing** (Are all three races equally strong?)



Balancing

- **Macromanagement:** Which units can be produced in a certain amount of time?
⇒ “ A Multi-objective Genetic Algorithm for Build Order Optimization in StarCraft II ” by Harald Köstler and Björn Gmeiner
- **Micromanagement:** Is it possible to predict which of two groups of units wins in combat?
⇒ No suitable approach for Starcraft II yet

Roadmap

- **Input:** Build Order, i.e. the list of units that have been produced until a certain point of time in the game
- **Goal:** Simulate and optimize the behavior (moving and attacking) of each single unit
 - ⇒ It can be predicted which player would succeed in a combat assuming optimal control

Overview

1 Motivation

2 Forward Simulation

3 Optimization

4 Results

Forward Simulation

- No freely available API for controlling units in the game directly
 - ⇒ An efficient forward simulation is required that determines the winner of an encounter based on a finite set of parameters
 - **Idea:** Describe the behavior of each unit by a number of parametrized **Potential Fields**
- ⇒ Units create multiple artificial potential fields around their position which are modeled as linear functions

Potential Fields

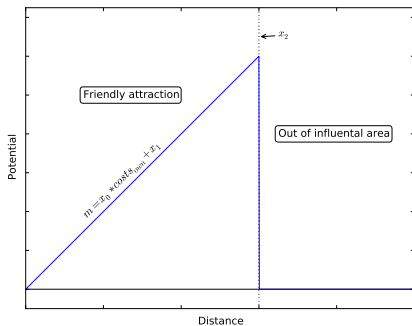


Figure 1 : Attractive potential of friendly units.

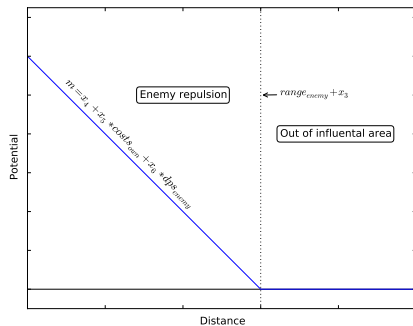


Figure 2 : Repulsive potential of enemy units.

Potential Fields

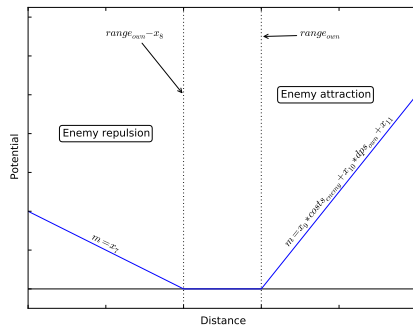


Figure 3 : Attractive potential of enemy units.

Forward Simulation

During a time step the following actions are performed by each unit:

- If **attacking** is possible, a target is chosen among all enemies within attack range by favoring units that can be defeated, prioritized by the amount of applicable damage
- Else, the unit **moves** and its position at the next time step is computed with the following equation:

$$p_{i+1}^{\vec{}} = \vec{p}_i + \vec{F} \times s$$

where \vec{p}_i is the position at time step i , \vec{F} the current force and s the movement range.

Forward Simulation

- The **force** of each unit is recomputed after a fixed number of time steps by accumulating the gradients of all potential fields applying to it:

$$\vec{F} = \sum_{j=1}^n \vec{F}_j$$

where n is the number of potential fields affecting the unit and \vec{F}_j the gradient of the j th potential field

- The forward simulation finishes when either all units of a player have been defeated or the simulation's duration exceeds a certain limit.

Overview

1 Motivation

2 Forward Simulation

3 Optimization

4 Results

Optimization

- **Goal:** Iteratively optimize the parameters for both opponents' units against each other
 - **Challenges:**
 - Large search space (at least 14 real valued parameters for each different type of unit)
 - No knowledge about the relationship between in- and output
- ⇒ **Genetic Algorithms** are suitable search heuristics for problems of this type

Optimization

General Procedure

- Encode the parameters of both opponents
- Choose suitable starting values for the optimization objective (strategy used by the opponent) for both populations
- Replace the objectives every n generation by the respective optima and reevaluate both populations
- The obtained results can be used to evaluate the effectiveness of both build orders against the respective other one

Single-Objective Genetic Algorithm

- The fitness of each individual is approximated with a simple formula:

$$\text{Fitness} = \text{total applied damage} + \text{total remaining health} + \text{value of killed units} + \text{value of remaining units}$$

⇒ Determine the **optimal combination of genetic operators** for the computational costlier multi-objective optimization

Multi-Objective Genetic Algorithm

- Considers all relevant objectives (applied damage, remaining health, value of units killed, value of units remaining etc.) independently to achieve a better spread of solutions
 - Based on **nondominated sorting** (NSGA-II)
- ⇒ Perform the actual optimization with suitable operators to **evaluate the effectiveness of certain build orders** compared to each other

Overview

1 Motivation

2 Forward Simulation

3 Optimization

4 Results

The End