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

Jonas Schmitt

July 2, 2015

Outline

Overview

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 (price pools up to US$170,000)
  $\Rightarrow$ 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
  $\Rightarrow$ It can be predicted which player would succeed in a combat assuming optimal control

Overview

Forward Simulation

- No freely available API for controlling units in the game directly
  $\Rightarrow$ 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

$\Rightarrow$ Units create multiple artificial potential fields around their position which are modeled as linear functions

Potential Fields

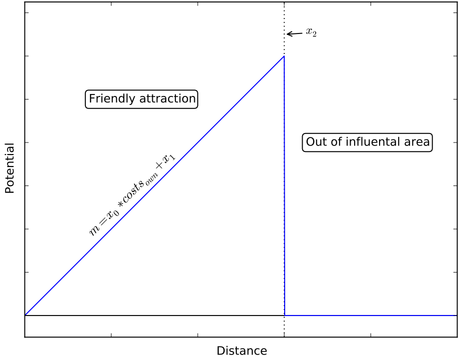

Figure 1 : Attractive potential of friendly units.

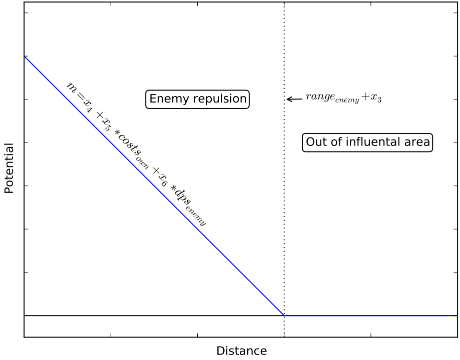

Figure 2 : Repulsive potential of enemy units.

Potential Fields

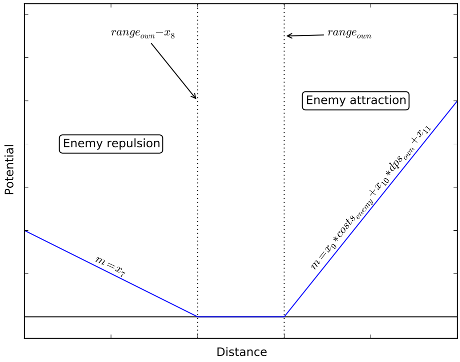

Figure 3 : Attractive potential of enemy units.

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:

  $$\vec{p_{i+1}} = \vec{p_i} + \vec{F} \times s$$

  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 jth 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

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

$\Rightarrow$ Genetic Algorithms are suitable search heuristics for problems of this type

Optimization

Overview

- 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:

$$\begin{aligned} \text{Fitness} &= \text{total applied damage} + \text{total remaining health} \\ &\quad + \text{value of killed units} + \text{value of remaining units} \end{aligned}$$

$\Rightarrow$ 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)

$\Rightarrow$ Perform the actual optimization with suitable operators to evaluate the effectiveness of certain build orders compared to each other

Overview

Results

- Measuring the Performance of different Combinations of common Genetic Operators (Selection, Crossover and Mutation Methods) for the single-objective genetic algorithm
- Online Performance: Average fitness of all evaluations
- Offline Performance: Average fitness of the optima of each generation
- Parameters used:

Population Size	Iterations	Generations per Iteration	Mutation Probability
500	100	10	1 %

Results

Selection	Crossover	Mutation	Online Performance
RWS	NPC	IM	82.17
SUS	SPC	IM	80.64
RWS	TPC	RM	79.19
RWS	UC	RM	79.12
SUS	SPC	BFM	79.04

Table 1 : Online Performance

SUS: Stochastic Universal Sampling, RWS: Roulette Wheel Selection, TS: Tournament Selection (Binary)

SPC: Single-Point Crossover, TPC: Two-Point Crossover, NPC: N-Point-Crossover (with $n = \frac{1}{2}$ number of parameters), UC: Uniform Crossover

BFM: Bit-Flip Mutation, IM: Interchanging Mutation, RM: Reversing Mutation

Results

Selection	Crossover	Mutation	Offline Performance
SUS	SPC	IM	94.50
RWS	NPC	IM	94.09
RWS	TPC	RM	93.78
TS	TPC	IM	93.52
RWS	TPC	IM	93.48

Table 2 : Offline Performance

SUS: Stochastic Universal Sampling, RWS: Roulette Wheel Selection, TS: Tournament Selection (Binary)
SPC: Single-Point Crossover, TPC: Two-Point Crossover, NPC: N-Point-Crossover (with $N = \frac{1}{2}$ number of parameters), UC: Uniform Crossover
BFM: Bit-Flip Mutation, IM: Interchanging Mutation, RM: Reversing Mutation

Thank you for the attention!