Optimization in StarCraft: *Building an Effective Army*

Jessica Garrett
Math 3301; Fall '14

Background

- RTS game publish by Blizzard Entertainment in '98
- Extremely strategic in design
- Requires quick risk assessment skills
- Goal - defeat your opponent's army before he/she defeats yours
- *Minerals*, *vespene gas* and supply must be gathered/built
- Infantry units have various unique parameters

Problem Description

The objective in this project is to build the most effective army given a multitude of constraints. We are defining *effective* as overall *damage* and *life* of the entire army. The decision variable is the number of each type of infantry unit to produce. Constraints bounding this optimization problem are time, total mineral and vespene count, supply and army variety, as we want an assortment of units.

Mathematical Model

Variables:
A_i = # of units produced
D_i= Damage
L_i= Life
S_i= Supply
M_i= Mineral
V_i= Vespene
T_i= Build time
Supply = Total supply
Mineral = Total mineral
Vespene = Total vespene
Time = Max time

Constraints:
$\sum A_i S_i \leq Supply$
$\sum A_i M_i \leq Mineral$
$\sum A_i V_i \leq Vespene$
$\sum A_i T_i \leq Time$
$\sum A_{command} \geq 1$
$\sum A_{barracks} \geq 1$
$\sum A_{factory} \geq 1$

Objective Function:
Max $\sum A_i (D_i + L_i)$

Resources:

- modeling software AMPL
- statistics – us.battle.net/sc2/en/game.

Assumptions

- Only ground units will be considered for production.
- The model will only include non-upgraded units.
- All necessary production buildings (with add-ons) have already been built at the beginning of the simulation.
- The element of time is not considered as a continuous domain.

AMPL Model

```
set UNITS;                              #infantry units

param mineral {UNITS} > 0;              # mineral cost for each unit
param vespene {UNITS} >= 0;             # vespene cost for each unit
param supply {UNITS} > 0;               # supply count for each unit
param buildtime {UNITS} > 0;            # production time in seconds
param damage {UNITS} > 0;               # damage associated for each unit
param life {UNITS} > 0;                 # amount of life for each unit

param maxTime > 0;                      # total time allotted for
production
param maxMineral > 0;                   # total mineral
param maxVespene > 0;                   # total vespene
param maxSupply > 0;                    # total supply

var Make {u in UNITS} >= 0 integer;     # units to produce
#Objective: max life and damage (best army)
maximize Army: sum {u in UNITS} damage[u] * Make[u]
                        + sum {u in UNITS} life[u] * Make[u];
#Production Constraints
subject to Time: sum {u in UNITS} buildtime[u] * Make[u] <= maxTime;
subject to Mineral: sum {u in UNITS} mineral[u] * Make[u] <= maxMineral;
subject to Vespene: sum {u in UNITS} vespene[u] * Make[u] <= maxVespene;
subject to Supply: sum {u in UNITS} supply[u] * Make[u] <= maxSupply;

#Variety Constraints (easily assigned by user according to
build/opponent):
subject to SCV: Make['Scv'] =3;
subject to Marine:  Make['Mare'] >=4;
subject to Marrader:  Make['Marr'] >=4;
#subject to Reaper:  Make['Reap'] >=1;
#subject to Ghost:  Make['Ghst'] >=1;
#subject to Hellion:  Make['Hell'] >=1;
subject to SeigeTank:  Make['Sgtk'] >=1;
#subject to Thors: Make['Thor'] >=1;
```

AMPL Data

```
data;

set UNITS := Scv Mare Marr Reap Ghst Hell Sgtk Thor;

param:        mineral vespene supply  buildtime damage life :=

#Command Center:
  Scv               50       0      1       17      5      45

#Barracks:
 Mare               50       0      1       25      6      45
 Marr               100      25     2       30      10     125
 Reap               50       50     1       45      4      60
 Ghst               200      100    2       40      10     100

#Factory:
 Hell               100      0      2       30      8      90
 Sgtk               150      125    3       45      15     160
 Thor               300      200    6       60      30     400;

param maxTime := 10000;
param maxMineral := 1500;
param maxVespene := 500;
param maxSupply := 70;
```

AMPL Output / Results

```
CPLEX 12.6.0.1: optimal
integer solution;
objective 1904
1 MIP simplex iterations
0 branch-and-bound nodes
```

```
Make [*] :=
Ghst  0
Hell  0
Mare  4
Marr  7
Reap  0
 Scv  3
Sgtk  1
Thor  1
```

The only associated slack was with buildtime as Time = 466. All other constraints were used to the max. In general, Marraders were selected because they have the best cost/benefit ratio.

Future Works

- Implement time as a continuous variable
- Consider building and upgrade costs
- Include all possible units