State Evaluation and Opponent Modelling in Real-Time Strategy Games

Graham Erickson

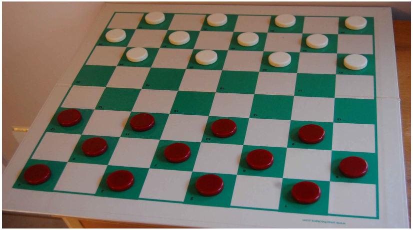
University of Alberta

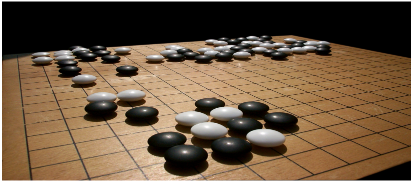
September 23, 2014

Layout

Game AI

- Enabling computers to play games





AI for Video Games

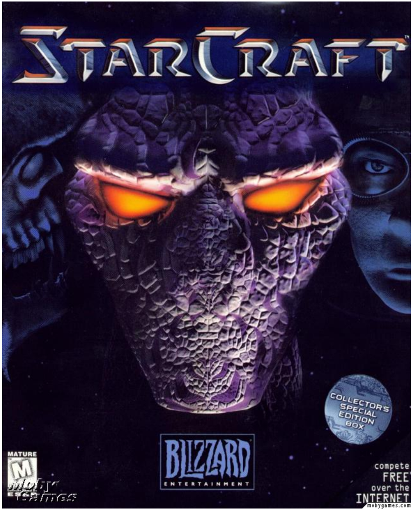
- Why work on video games?
- Tools for balancing
- More interesting opponents
- Dynamic game elements

Real-Time Strategy

- Combat simulation games
- Manage resources, build units, engage in battles
- Simultaneous moves
- Real-time
- Imperfect information

StarCraft

- Blizzard Entertainment
- 1998
- Commercial and critical success
- Three factions
  - Protoss
  - Terran
  - Zerg
- Known to be well-balanced



StarCraft AI

- Why StarCraft?
- Large online community
- Professional players
- Replays from various ladders freely available
- BWAPI (Brood War API)
  - C++
  - Can test against programs and humans

StarCraft AI Tournaments

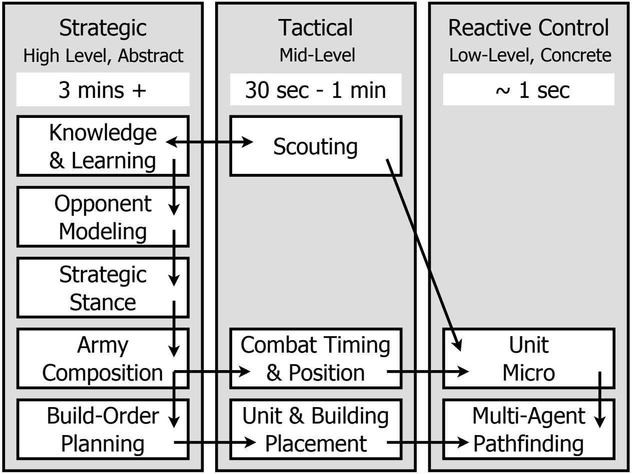
- AI against AI
- Nowhere near human skill
- AIIDE
- CIG
- SSCAI
- UAlbertaBot won in 2013

Artificial Intelligence and Interactive Digital Entertainment

Terminology

- Macro
  - Managing resources
  - Build-orders
  - High-level plans
- Micro
  - Controlling units in battles
  - Path finding

Strategy


Strategic
High Level, Abstract
3 mins +
Knowledge & Learning
Opponent Modeling
Strategic Stance
Army Composition
Build-Order Planning
Tactical
Mid-Level
30 sec - 1 min
Scouting
Combat Timing & Position
Unit & Building Placement
Reactive Control
Low-Level, Concrete
~ 1 sec
Unit Micro
Multi-Agent Pathfinding

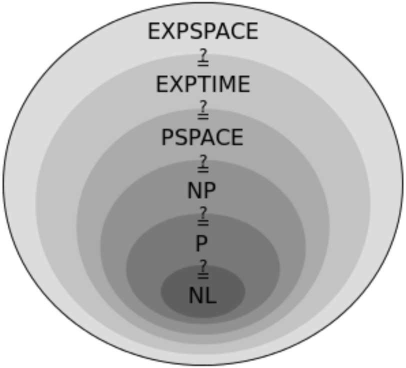
[1]

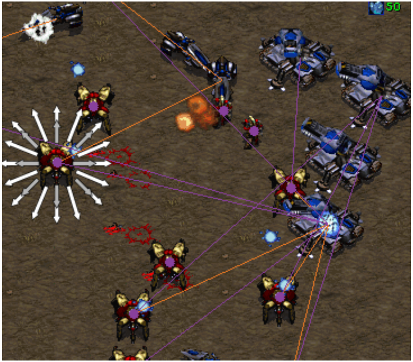
Why study RTS?

- Well-defined environments
- Can be broken into sub-tasks
- Areas in AI [2]
  - Adversarial planning under uncertainty
  - Learning and opponent modeling
  - Spatial and temporal reasoning
- Abstraction is necessary

Complexity of shooting game

- PSPACE-hard [3]





Layout

What is a Build-Order?

- Sequence
- Embody high-level strategy
- String of characters

TIME	ACTION	SUPPLY
00:00	SCV	7/11
00:18	SCV	8/11
00:35	SCV	9/11
00:52	SCV	10/11
01:08	Barracks	10/11
01:22	Refinery	10/11
01:28	SCV	11/11
01:51	Supply Depot	11/11
02:17	Tech Lab	11/11
02:21	SCV	12/19
02:38	SCV	13/19
02:47	Reaper	14/19

Why Cluster Build-Orders?

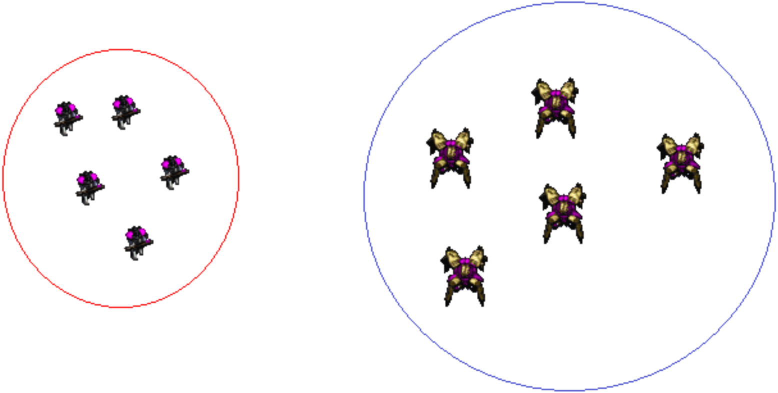
- Hard coded rules
  - Expert knowledge
- Game balancing is an extensive process
  - Starcraft was patched continuously for 11 years

Why Cluster Build-Orders?

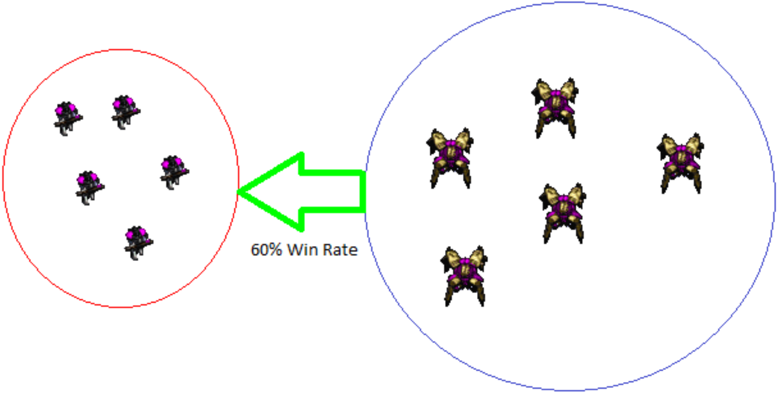
- novel strategies
- avoid expert knowledge
- empirical basis for responses

Why Cluster Build-Orders?

Why Cluster Build-Orders?



Why Cluster Build-Orders?



Jeff Long's Master's Thesis

- 100 WarCraft III replays [4]
- Hand-labeled build-orders
- Classification problem
- Sequence alignment
- Populate payoff matrices

Sequence Alignment

abba

ba

Sequence Alignment

abba

_b_a

Sequence Alignment

$$\sum_{i=0}^{n} S(A_i, B_i)$$

Needleman-Wunsch Sequence Alignment

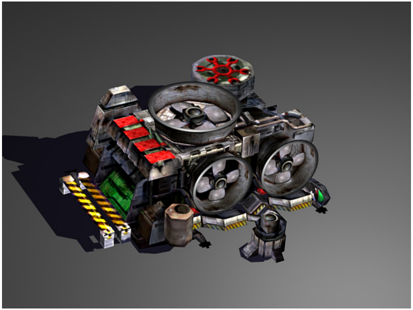
- Dynamic greedy algorithm
- $O(nm)$
- Score and alignment

Edit or Levenshtein Distance

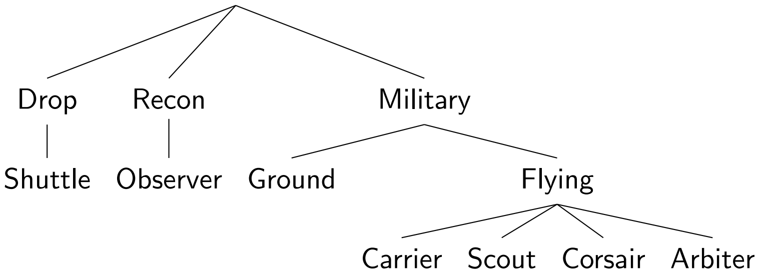
$$S(a, b) = \begin{cases} 0 & \text{if } a = b \\ -1 & \text{if } a \neq b \end{cases}$$

Sequence Alignment for StarCraft

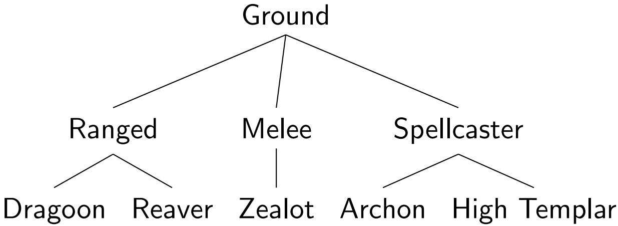
- Unit Similarity
- We introduce supply
- Strongly reward matches
- Additional penalties for mis-matches



Mis-Match Penalty I



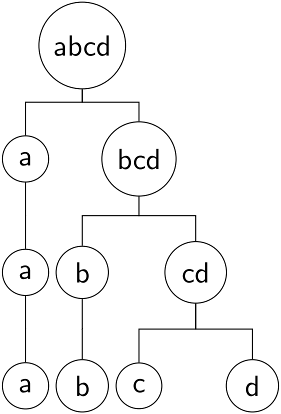
Mis-Match Penalty II



Clustering

- Now that we have a similarity metric we can cluster build-orders
- We want a technique that works using just a similarity matrix

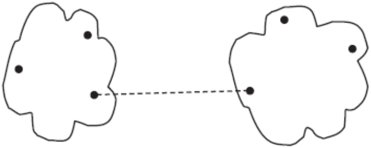
Hierarchical Clustering



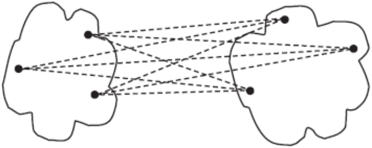
Agglomerative Hierarchical Clustering

Input: Similarity matrix S
Compute proximity matrix P from S
while $|P| > 1$ **do**
    Merge clusters i and j where P_{ij} is maximized
    Update P
end while

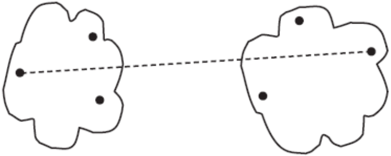
Proximity Measures



(a) MIN (single link.)



(c) Group average.



(b) MAX (complete link.)

Proximity Measures

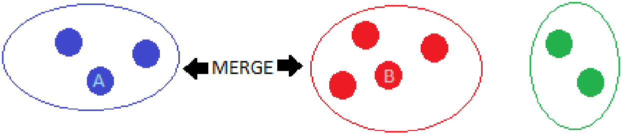
$$Max(C_1, C_2) = max(\{S_{ij} | i \in C_1, j \in C_2\})$$

$$Min(C_1, C_2) = min(\{S_{ij} | i \in C_1, j \in C_2\})$$

$$Average(C_1, C_2) = \frac{\sum_{i \in C_1} \sum_{j \in C_2} S_{ij}}{|C_1| * |C_2|}$$

Choosing a Proximity Measure

- CoPhenetic Correlation Coefficient



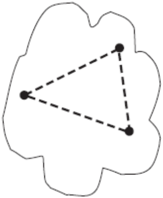
Data

- Data collected by Gabriel Synnaeve [5]
- Protoss versus Protoss (∼400 games)
- Protoss versus Terran (∼2000 games)
- Replays are taken from major amateur ladders
- Replays are already parsed

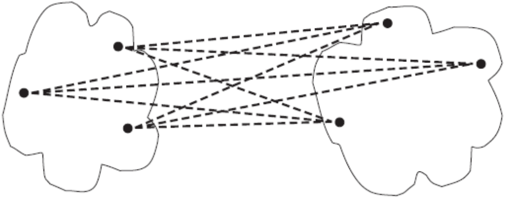
CPCC values for PvT

Linkage Policy	Protoss CPCC	Terran CPCC
Min	0.68256	0.77136
Max	0.18612	0.16551
Average	0.83518	0.85562
Ward	0.61552	0.54474

Cohesion and Separation

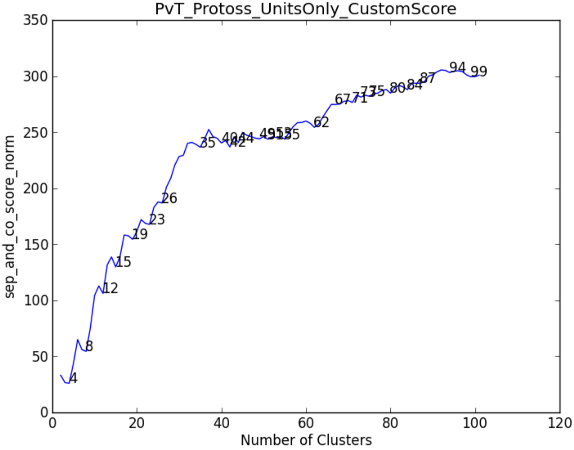


(a) Cohesion.



(b) Separation.

Choosing Partitional Clustering PvT Protoss



Protoss Clusters

- Cluster 1 (*rush*)
  - Short
  - Zealots and Dragoons
  - Probably rushes
- Cluster 2 (*drop*)
  - Small
  - Reaver drops
- Cluster 3 (*drag*)
  - Mid-length
  - Dragoons
- Cluster 4 (*big*)
  - Very large
  - Tough to see high-level coherence

	rush	*tanks*	*drop*	*big*
rush	0.07 (15)	-0.09 (33)	0 (0)	-1.0 (1)
drop	0 (0)	1.0 (2)	-1.0 (1)	0.33 (3)
drag	0.5 (4)	0.30 (158)	0.11 (9)	-0.03 (203)
big	0 (0)	1.0 (4)	1.0 (1)	0.17 (1655)

Terran Clusters

- Cluster 1 (*rush*)
  - Short
  - Marines
- Cluster 2 (*tanks*)
  - Mid-length
  - Marines $\rightarrow$ Vultures and Tanks or just Tanks
- Cluster 3 (*drop*)
  - Varying lengths
  - Goliaths and Dropships
- Cluster 4 (*big*)
  - Very large
  - Tough to see high-level coherence

	rush	*tanks*	*drop*	*big*
rush	0.07 (15)	-0.09 (33)	0 (0)	-1.0 (1)
drop	0 (0)	1.0 (2)	-1.0 (1)	0.33 (3)
drag	0.5 (4)	0.30 (158)	0.11 (9)	-0.03 (203)
big	0 (0)	1.0 (4)	1.0 (1)	0.17 (1655)

Conclusion

- Build-orders are just sequences of characters
- Sequence alignment for developing similarity metrics
- Agglomerative hierarchical clustering
- Clusters show some cognitive coherence
- Future work
  - Different clustering techniques
  - Experimenting with custom cost functions
  - Using payoff matrices to influence in-game decision making

Layout

1. Introduction
2. Build-Order Clustering
3. State Evaluation

Problem

- Given a state, predict the winner
- Perfect information
- Identify important features
- Estimate player skill

Motivation

- Search algorithms that require evaluation have had success in other games
- Part of a research initiative into hierarchical search systems [6]
- Used for pruning and rule-based decision making

Objectives

- Present a feature set
- Micro skill estimation metric
- Show effectiveness of technique across time-steps

Data

- Synnaeve data-set
- Protoss versus Protoss
- Parser developed
  - Some errors with destruction events
  - Control over battle detection

SCFeatureExtractor

- https://github.com/gkericks/SCFeatureExtractor
- C++
- BWAPI
- Computes feature vectors and writes them to file periodically

Battles

- Isolated skirmishes
- Micro game

Battle Extraction

- Identify battles
- Log unit info when the battle starts
  - Time
  - Health
  - Location
- Let units enter at different times
- Battles time out or end by rout

Battle Example I

Battle Example II

Battle Example III

Battle Example IV

Battle Example V

Battle Example VI

Battle Example VII

Battle Example VIII

Battle Example IX

Battle Example X

Battle Example XI

Battle Example XII

Battle Example XIII

Features

- Feature vectors are extracted every 10 seconds
- Feature values are in terms of differences
  - Player A has D_A Dragoons
  - Player B has D_B Dragoons
  - Feature is $D_A - D_B$
- Two feature vectors are added for each state
  - Symmetric match-up

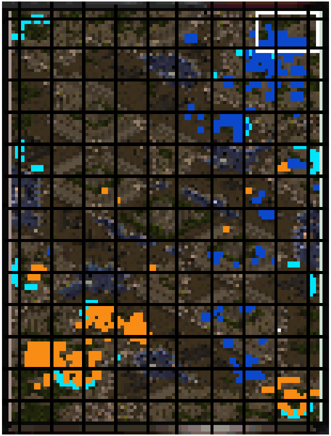
Economic Features

- Average unspent resources
- Income

Military Features

- Unit and building counts
- Units that are ammo not included
- Research/Upgrades not included

Map Coverage Feature



Micro Skill Estimation

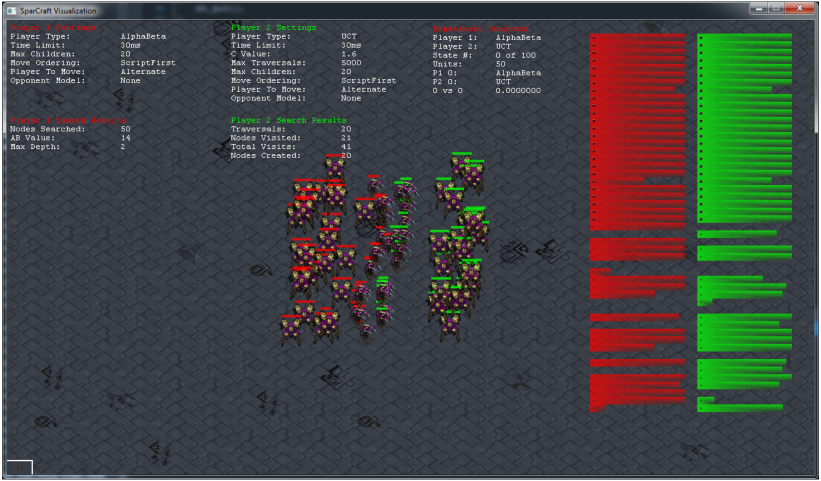
- Combat game
- Specific type of skill
- Skill estimate can be used as a feature

Baseline

- Work done in Poker [7]
- Play out same situation using a baseline player
- Compare agent and baseline outcome

SparCraft

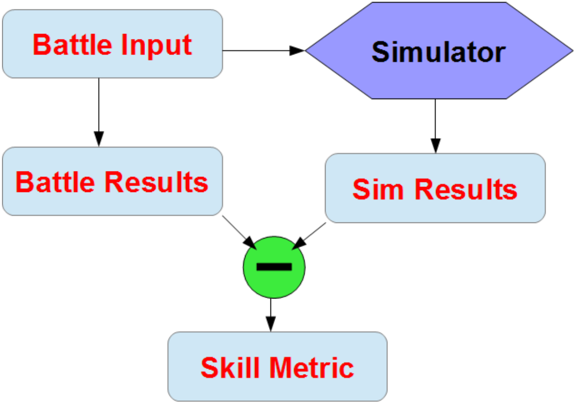
- https://code.google.com/p/sparcraft/



Scripted Player

- No-OverKill Attack-Value (*NOK-AV*)
- Targets highest damage-per-frame to hit-point ratio
- Buildings are just obstacles

Battle Skill Metric

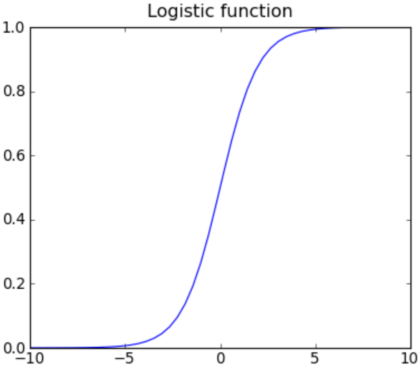


Macro Skill

- High-level decision making
- Number of frames that supply is maxed out for
- Number of idle production facilities (PF)
- Number of units queued (Q)

Learning

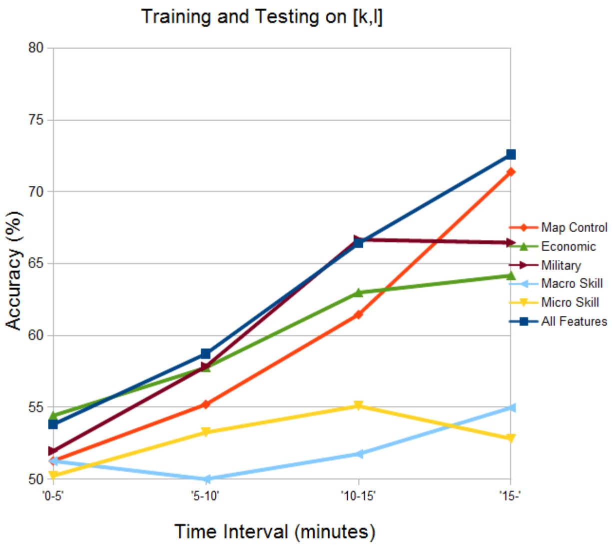
- Logistic Regression
- Learn feature weights



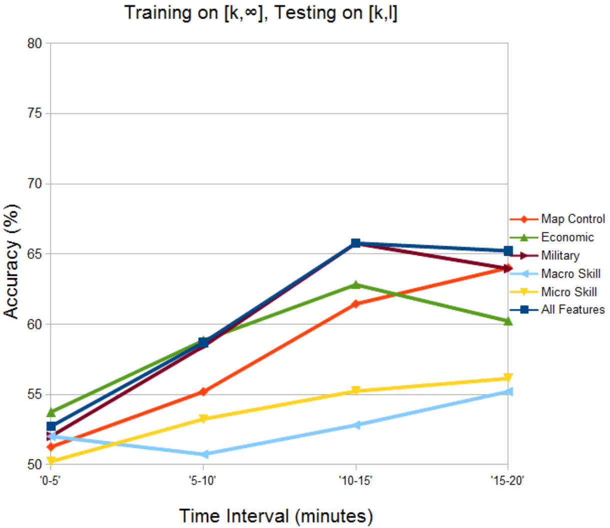
Breakdown of Time Intervals

Time (min)	Games	Examples
0-5	391	23418
5-10	386	22616
10-15	364	19836
15-20	289	14996
20-	211	31060

Feature Set Evaluation



With Larger Training Sets



Problems with Micro Skill Estimator

- Low number of repeat players in data-set
- Late game units
- No external ranking

2013 AIIDE StarCraft AI Tournament

- 8 AI systems
- 10 maps
- Each bot plays each other bot 20 times on each map
- Ranked by win percentage

Ranking from AIIDE 2013 StarCraft Competition

UAlbertaBot	82.43%
Skynet	72.77%
Aiur	60.29%
Ximp	55.29%
Xelnaga	49.96%
ICEStarCraft	47.82%
Nova	27.47%
BTHAI	3.93%

Ranking using Micro Skill Averaged

Nova	7.65
UAlbertaBot	3.30
Aiur	1.01
ICEStarCraft	-0.026
Ximp	-1.91
BTHAI	-3.03
Skynet	-4.61
Xelnaga	-5.60

Ranking using Micro Skill with Variance Control

Nova	7.59
UAlbertaBot	1.97
Aiur	0.85
ICEStarCraft	0.01
Ximp	-1.79
Xelnaga	-2.99
BTHAI	-3.13
Skynet	-4.51

Conclusion

- Predict game outcome
- Noisy problem
- Feature set has $> 70\%$ accuracy in the later stages of a match
- Average unspent resources
- Income
- Map control
- Skill estimation through simulation

How hard is RTS?

- Huge state space
  - Average map size is 128*128 tiles
  - Approx. 50 unit types
  - Approx. 200 units
  - 10^{1685} possible unit locations
  - Ignoring Health, Attacks, Resources etc.

Sequence Alignment

- Needleman-Wunsch sequence alignment algorithm [8]
- Sequences A and B can have gaps inserted to make aligned sequences A' and B'
- Take $S(a, b)$ to be the similarity between two characters a and b
- Take $S(-, a)$ to be the *gap penalty* for some character a
- Maximize alignment score:

$$\sum_{i=0}^{n} S(A'_i, B'_i)$$

Example

abba

ba

$$S(a,b) = \begin{cases} 0 & \text{if } a = b \\ -1 & \text{if } a \neq b \end{cases}$$

abba

_b_a

$S(a, b)$ is the Levenshtein or edit distance [9]

Needleman-Wunsch Sequence Alignment

- Dynamic program
- Populates a matrix M
- M_{ij} is the score of an optimal alignment between the first i characters in A and the first j characters in B

-	-	A	T	C	G	A	C
-	0	-4	-8	-12	-16	-20	-24
C	-4	-3	-7	-3	-7	-11	-15
A	-8	1	-3	-7	-6	-2	-6
T	-12	-3	6	2	-2	-6	-5
A	-16	-7	2	3	-1	3	-1
C	-20	-11	-2	-1	0	-1	8

Needleman-Wunsch Initialize M

$T = 0$
for $i \in [1...n]$ **do**
    $M_{i0} = T + S(-, A_i)$
    $T = M_{i0}$
end for
$T = 0$
for $j \in [1...m]$ **do**
    $M_{0j} = T + S(-, B_j)$
    $T = M_{0j}$
end for
$M_{00} = 0$

Needleman-Wunsch Dynamic Program

for $i \in [1...n]$ **do**
  for $j \in [1...m]$ **do**
    $match = M_{i-1,j-1} + S(A_i, B_j)$
    $gapA = M_{i-1,j} + S(-, A_i)$
    $gapB = M_{i,j-1} + S(-, B_j)$
    $M_{i,j} = max(match, gapA, gapB)$
  end for
end for

Similarity Metric

- Goal is to cluster sequences
- In [10] sequence alignment is used to define sequence similarity

$$dis(A, B) = M_{0,0}$$

$$S'(a, b) = \begin{cases} S(a, b) & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

$$dis_{correct}(A, B) = \sum_{i=0}^{n} S'(A_i, B_i)$$

$$Sim_{align}(A, B) = dis(A, B)/dis_{correct}(A, B)$$

Similarity Metric

$$cor(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

$$Num\_Correct(A,B) = \sum_{i=0}^{n} cor(A_i, B_i)$$

$$Sim_{Significance}(A,B) = Num\_Correct(A,B)/n$$

$$Sim(A,B) = Sim_{align}(A,B) * Sim_{Significance}(A,B)$$

General Cost Function

$$\rho(a, b) = \begin{cases} k * c(a) & \text{if } a == b \\ |c(a) - c(b)| - \phi(a, b) & \text{otherwise} \end{cases}$$

- c is some attribute of a unit type
- k is a constant
- ϕ is a mis-match penalty.

Unit Similarity

- Recall that Needleman-Wunsch uses a character similarity function $S(a, b)$
- Also called a cost function
- More interesting to design a domain specific cost function
- Let ρ be a custom cost function of character a and b

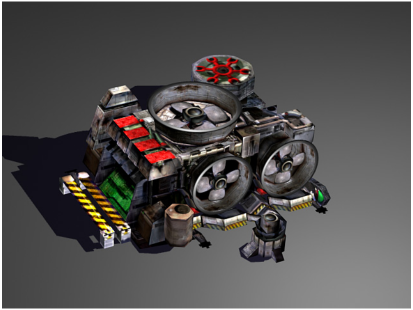
General Cost Function

$$\rho(a,b) = \begin{cases} k * c(a) & \text{if } a == b \\ |c(a) - c(b)| - \phi(a,b) & \text{otherwise} \end{cases}$$

- c is some attribute of a unit type
- k is a constant
- ϕ is a mis-match penalty.

Cost Function for StarCraft

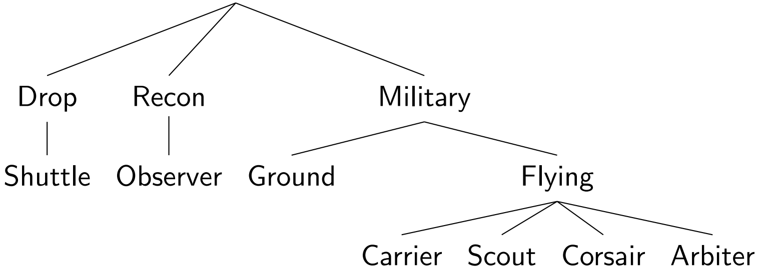
- For c we introduce supply
- Supply is for limiting unit counts
- Jeff Long used $k = 16$
- Strongly reward matches



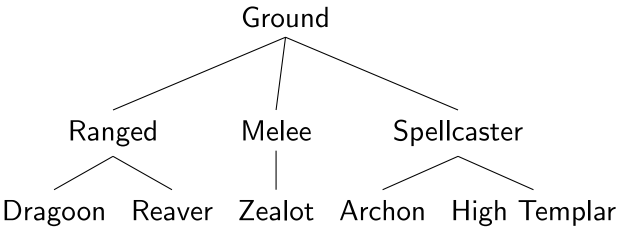
Mis-Match Penalty

- Defining $\phi(a, b)$
- Larger when a and b are more different
- We propose a unit *Ontology*
    - Hierarchical Categorization
    - Penalties are assigned depending on at what level the units differ
    - More essential differences have higher penalties

Protoss Ontology I



Protoss Ontology II



Cluster Evaluation

- What clusterings should we choose?

Choosing a Cluster Proximity Measure

- Recall: Agglomerative Hierarchical clustering requires a proximity measure
- How to choose the best one for the data?

CoPhenetic Correlation Coefficient

- Similarity matrix S and proximity matrix P
- During clustering there will be a iteration where two elements x and y are first members of the same cluster
- The proximity of the two clusters at that iteration is the CoPhenetic distance for x and y
- Populate a matrix P' of CoPhenetic distances
- CPCC is correlation between P' and S

CPCC values for PvP data

Linkage Policy	CPCC
Min	0.62337
Max	0.21094
Average	0.76905
Ward	0.56441

CPCC values for PvT

Linkage Policy	Protoss CPCC	Terran CPCC
Min	0.68256	0.77136
Max	0.18612	0.16551
Average	0.83518	0.85562
Ward	0.61552	0.54474

Choosing a Partitional Clustering

- Have a metric for a partitional clustering [11]
- Chose the clustering that optimizes the metric

Cohesion

$$Cohesion(C) = \sum_{i \in C} \sum_{j \in C} S_{i,j}$$

Separation

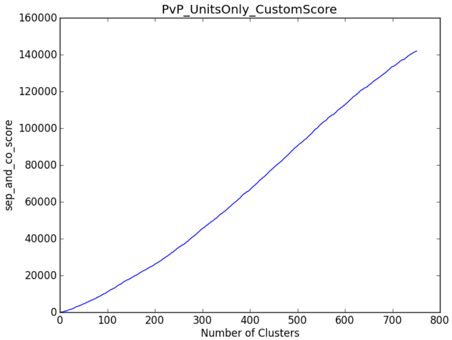
$$Sep(C, C') = \sum_{i \in C} \sum_{j \in C'} S_{i,j}$$

$$Separation(C) = \sum_{\substack{C' \in \kappa \\ C' \neq C}} Sep(C, C')$$

Combining Cohesion and Separation

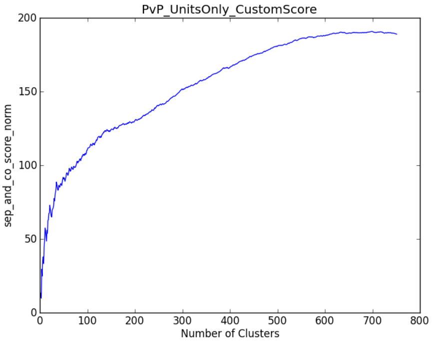
$$Sep\_and\_Co(\kappa) = \sum_{C \in \kappa} \frac{Separation(C)}{Cohesion(C)}$$

Choosing Partitional Clustering PvP I



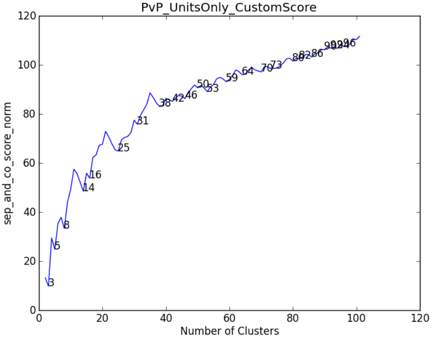
Sep_and_Co versus the number of clusters for the hierarchical clustering of the PvP dataset

Choosing Partitional Clustering PvP II



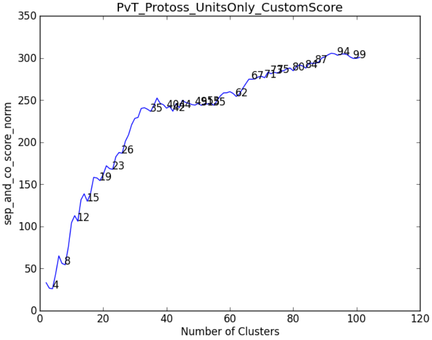
Sep_and_Co versus the number of clusters for the hierarchical clustering of the PvP dataset normalized by number of clusters

Choosing Partitional Clustering PvP III



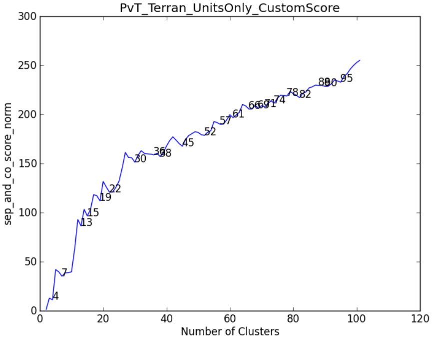
Sep_and_Co versus the number of clusters for the hierarchical clustering of the PvP dataset normalized by number of clusters on the domain of [2,100]

Choosing Partitional Clustering PvT Protoss



Sep_and_Co versus the number of clusters for the hierarchical clustering of the PvT dataset normalized by number of clusters on the domain of [2,100] just using Protoss players

Choosing Partitional Clustering PvT Terran



Sep_and_Co versus the number of clusters for the hierarchical clustering of the PvT dataset normalized by number of clusters on the domain of [2,100] just using Terran players

Building Payoff Matrices I

- Quantify how build-orders in each cluster preform against each other
- Game balance
- Strategy response
- Populated via replay data

Building Payoff Matrices II

$$G_{ij} = \frac{w_{ij} - l_{ij}}{t_{ij}}$$

- i is the row player
- j is the column player
- w_{ij} is the number of wins for cluster i against cluster j
- l_{ij} is the number of losses for cluster i
- $t_{ij} = w_{ij} + l_{ij}$

Payoff Matrix from PvP Data

	1	2	3
1	0 (0)	-1.0 (1)	0 (0)
2	1.0 (1)	0.0 (640)	0.2 (25)
3	0 (0)	-0.2 (25)	0 (60)

Problems

- Diagonal is uninteresting
  - PvP is a symmetric match-up
- Diagonal has most of the examples
- Cluster 1 is very small

Payoff Matrix from PvT I

	1	2	3	4
1	0 (2)	0 (0)	1.0 (2)	0 (0)
2	1.0 (1)	-1.0 (1)	0 (0)	0 (0)
3	-1.0 (1)	0 (0)	-0.09 (33)	0.07 (15)
4	0.15 (1858)	0.2 (10)	0.32 (60)	0.5 (4)

Payoff Matrix from PvT I

	1	2	3	4
1	0 (2)	0 (0)	1.0 (2)	0 (0)
2	1.0 (1)	-1.0 (1)	0 (0)	0 (0)
3	-1.0 (1)	0 (0)	-0.09 (33)	0.07 (15)
4	0.15 (1858)	0.2 (10)	0.32 (60)	0.5 (4)

- There are still small clusters!
- These might not represent a coherent strategy

Removing Small Clusters

- Let C be the topmost cluster of the hierarchical clustering
- Let T be a threshold size
- Let P be kept clusters
- Split C into two clusters
- Larger cluster is added to P
- If smaller cluster is $\geq T$ it is added to P, discarded otherwise

Synnaeve's Extraction Algorithm

- Similar
- No buildings
- Starts with unit destruction events
  - Ours start with attacks
- Only start and end timestamps
  - Ours has timestamps for when units enter the battle

Preprocessing

- Some replays contained strange activity
  - Giving up when clearly ahead
  - AFK
- Winner not always clearly marked

Determining the Winner

- Flags
  - *isWinner*
  - *playerLeft* (with time-stamp)
- If *isWinner* is present, use that
- Otherwise *playerLeft* and game score are used

Determining the Winner

- If no *playerLeft* flag
    - game score is used, if scores are close ($\Delta < T$) replay is discarded
- If one *playerLeft* flag
    - Opposite player is winner
    - Unless that conflicts with game score (type A)
- Two *playerLeft* flags
    - Player who left second is chosen
    - Unless that conflicts with game score (type B)

Breakdown of Discarded Replays

Games	Number of Games
Original	447
Kept	391
No Status Close Score	30
Conflict Type A	24
Conflict Type B	1
Corrupt	1

Economic Features

- Let R_{cur} be current unspent resources
- Let R_{tot} be current unspent resources
- Let T be current frame number
- Average unspent resources:

$$U = (\sum_{t \leq T} R_{cur})/T$$

- Income:

$$I = \frac{R_{tot}}{T}$$

Map Coverage Feature

- Map divided into grid (2-by-2 build tiles)
- Ratio of occupied to total tiles
- Units to walkable space

$$MC(p) = \sum_{pos \in P} f(pos, p)$$

$$f(pos, p) = \begin{cases} 1 & \text{if } pos \text{ is occupied by } p \\ 0 & \text{otherwise} \end{cases}$$

Battle Outcome

- Life-time damage:

$$\mathrm{LTD2}_{start}(U) = \sum_{u \in U} \sqrt{\mathrm{HP}(u)} \cdot \mathrm{DMG}(u)$$

- U is the set of units for a player
- Favours having multiple units to single units given equal summed health
- Rewards keeping units alive that can deal greater damage quicker.

Battle Outcome

- Units can enter battles at varying times
- Let T be the length of the battle
- Let $\mathrm{st}(u)$ be the time unit u entered the battle

$$\mathrm{LTD2}_{end}(U) = \sum_{u \in U} \frac{T - \mathrm{st}(u)}{T} \cdot \sqrt{\mathrm{HP}(u)} \cdot \mathrm{DMG}(u)$$

Battle Value

- P_s and O_s are unit sets for player and opponent at the start of the battle
- P_{out} and O_{out} are unit sets for player and opponent at the end of the battle

$$V^P = (\text{LTD2}_{end}(P_{out}) - \text{LTD2}_{end}(O_{out})) - (\text{LTD2}_{start}(P_s) - \text{LTD2}_{start}(O_s))$$

- P_β and O_β are the unit sets from the baseline player

$$V^\beta = (\text{LTD2}_{end}(P_\beta) - \text{LTD2}_{end}(O_\beta)) - (\text{LTD2}_{start}(P_s) - \text{LTD2}_{start}(O_s))$$

Battle Skill Metric

$$\beta_{tot} = \sum_{i=1}^{n} (V_i^P - V_i^\beta)$$

$$\beta_{avg} = \frac{\beta_{tot}}{n}$$

Battle Skill Metric

$$\beta_{var} = \frac{1}{n}\sum_{i=1}^{n}(V_i^P - \frac{\widehat{\text{Cov}}[V_i^P, V_i^\beta]}{\widehat{\text{Var}}[V_i^P]} \cdot V_i^\beta)$$

Macro Skill

- High-level decision making
- Number of frames that supply is maxed out for

$$SF = \sum_{t \leq T} f(t)$$

$$f(t) = \begin{cases} 1 & \text{if } S_{cur} = S_{max} \text{ at time t} \\ 0 & \text{otherwise} \end{cases}$$

- Number of idle production facilities (PF)
- Number of units queued (Q)

Learning

- Logistic Regression
- Matrix X with n examples (rows) and k features (columns)
- Corresponding response vector Y
- Gives k weights K such that

$$X \cdot K = Y'$$

$$T(g(Y')) \approx Y$$

$$g(s) = \frac{1}{1 + e^{-s}}$$

- T is a threshold function

Evaluation

- 10-fold cross validation by games
- Reporting accuracy
  - Proportion of correct predictions
  - Responses threshold at 0.5
- Average Log-Likelihood

$$L(y, r) = y \cdot \log(r) + (1 - y) \cdot \log(1 - r)$$

Feature Set Evaluation

Features	0-5	5-10
R_{cur}, I, U	54.42 (-0.686)	57.76 (-0.672)
UC	51.96 (-0.712)	57.84 (-0.682)
MC	51.27 (-0.693)	55.20 (-0.685)
β_{var}	50.23 (-0.693)	53.25 (-0.690)
SF, PF, Q	51.26 (-0.695)	49.96 (-0.695)
A	53.91 (-0.708)	58.81 (-0.680)
B	54.05 (-0.708)	58.66 (-0.681)
C	53.81 (-0.710)	58.72 (-0.681)

- A = economic/military features R_{cur}, I, U, UC
- B = A + map control feature MC
- C = B + skill features β_{var}, SF, PF, Q

Feature Set Evaluation

Features	10-15	15-
R_{cur}, I, U	62.98 (-0.647)	64.17 (-0.625)
UC	66.67 (-0.705)	66.46 (-0.644)
MC	61.45 (-0.657)	71.39 (-0.561)
β_{var}	55.09 (-0.690)	52.82 (-0.690)
SF, PF, Q	51.75 (-0.694)	54.97 (-0.709)
A	66.36 (-0.712)	69.22 (-0.613)
B	66.44 (-0.712)	69.87 (-0.608)
C	66.41 (-0.708)	72.59 (-0.587)

- A = economic/military features R_{cur}, I, U, UC
- B = A + map control feature MC
- C = B + skill features β_{var}, SF, PF, Q

With Larger Training Sets

Feature Set	0-5	5-10
R_{cur}, I, U	53.75 (-0.6875)	58.85 (-0.6708)
UC	52.03 (-0.6936)	58.43 (-0.6735)
MC	51.27 (-0.6943)	55.20 (-0.6872)
β_{var}	50.23 (-0.6931)	53.25 (-0.6896)
SF, PF, Q	52.02 (-0.6925)	50.74 (-0.6939)
A	53.19 (-0.6917)	58.74 (-0.6726)
B	52.60 (-0.6916)	58.56 (-0.6727)
C	52.73 (-0.6914)	58.70 (-0.6669)

- If time interval is $[k, l]$ training is done on examples in $[k, \infty)$ and tested on examples in $[k, l]$

With Larger Training Sets

Feature Set	10-15	15-20
R_{cur}, I, U	62.82 (-0.6510)	60.23 (-0.6562)
UC	65.76 (-0.6329)	63.96 (-0.6516)
MC	61.45 (-0.6588)	64.02 (-0.6385)
β_{var}	55.24 (-0.6899)	56.14 (-0.6868)
SF, PF, Q	52.82 (-0.6916)	55.21 (-0.6857)
A	65.28 (-0.6367)	63.58 (-0.6612)
B	64.89 (-0.6377)	63.99 (-0.6617)
C	65.77 (-0.6267)	65.23 (-0.6510)

- If time interval is $[k, l]$ training is done on examples in $[k, \infty)$ and tested on examples in $[k, l]$

Payoff Matrix from PvT II

	1	2	3	4
1	0.07 (15)	-0.09 (33)	0 (0)	-1.0 (1)
2	0 (0)	1.0 (2)	-1.0 (1)	0.33 (3)
3	0.5 (4)	0.30 (158)	0.11 (9)	-0.03 (203)
4	0 (0)	1.0 (4)	1.0 (1)	0.17 (1655)

Protoss Clusters

- Cluster 1
  - Short (in length) build-orders
  - Zealots and Dragoons
  - Probably rushes
- Cluster 2
  - Small
  - Scouts, Shuttles, Reavers, and Carriers
  - Reaver drops
- Cluster 3
  - Mid-length
  - Dragoons
- Cluster 4
  - Very Large
  - Tough to see high-level coherence

Terran Clusters

- Cluster 1
  - Short
  - Mostly just Marines
- Cluster 2
  - Mid-length
  - Start with Marines
  - Move to Vultures and Siege Tanks or just Siege Tanks
- Cluster 3
  - Varying lengths
  - Goliaths and Dropships
- Cluster 4
  - Long
  - Very large cluster
  - Tough to see high-level coherence

Bibliography I

- 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,” *TCIAIG*, 2013.
- M. Buro, “Real-time strategy games: A new AI research challenge,” in *IJCAI 2003*, pp. 1534–1535, International Joint Conferences on Artificial Intelligence, 2003.
- T. Furtak and M. Buro, “On the complexity of two-player attrition games played on graphs,” in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010* (G. M. Youngblood and V. Bulitko, eds.), (Stanford, California, USA), Oct. 2010.

Bibliography II

- J. Long, "Game theoretic and machine learning techniques for balancing games," Master's thesis, University of Saskatchewan, 2006.
- G. Synnaeve and P. Bessiere, "A dataset for StarCraft AI & an example of armies clustering," in *AIIDE Workshop on AI in Adversarial Real-time games 2012*, 2012.
- S. Ontanon, "Experiments with game tree search in real-time strategy games," *arXiv preprint arXiv:1208.1940*, 2012.

Bibliography III

- J. Davidson, C. Archibald, and M. Bowling, "Baseline: practical control variates for agent evaluation in zero-sum domains," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1005–1012, International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.
- V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, p. 707, 1966.

Bibliography IV

K. Duraiswamy and V. V. Mayil, "Similarity matrix based session clustering by sequence alignment using dynamic programming," *Computer and Information Science*, vol. 1, no. 3, p. P66, 2008.

P.-N. Tan, M. Steinbach, and V. Kumar, "Cluster analysis: basic concepts and algorithms," *Introduction to data mining*, pp. 487–568, 2006.