Bayesian Programming Applied to Starcraft

Micro-Management and Opening Recognition

Gabriel Synnaeve and Pierre Bessière

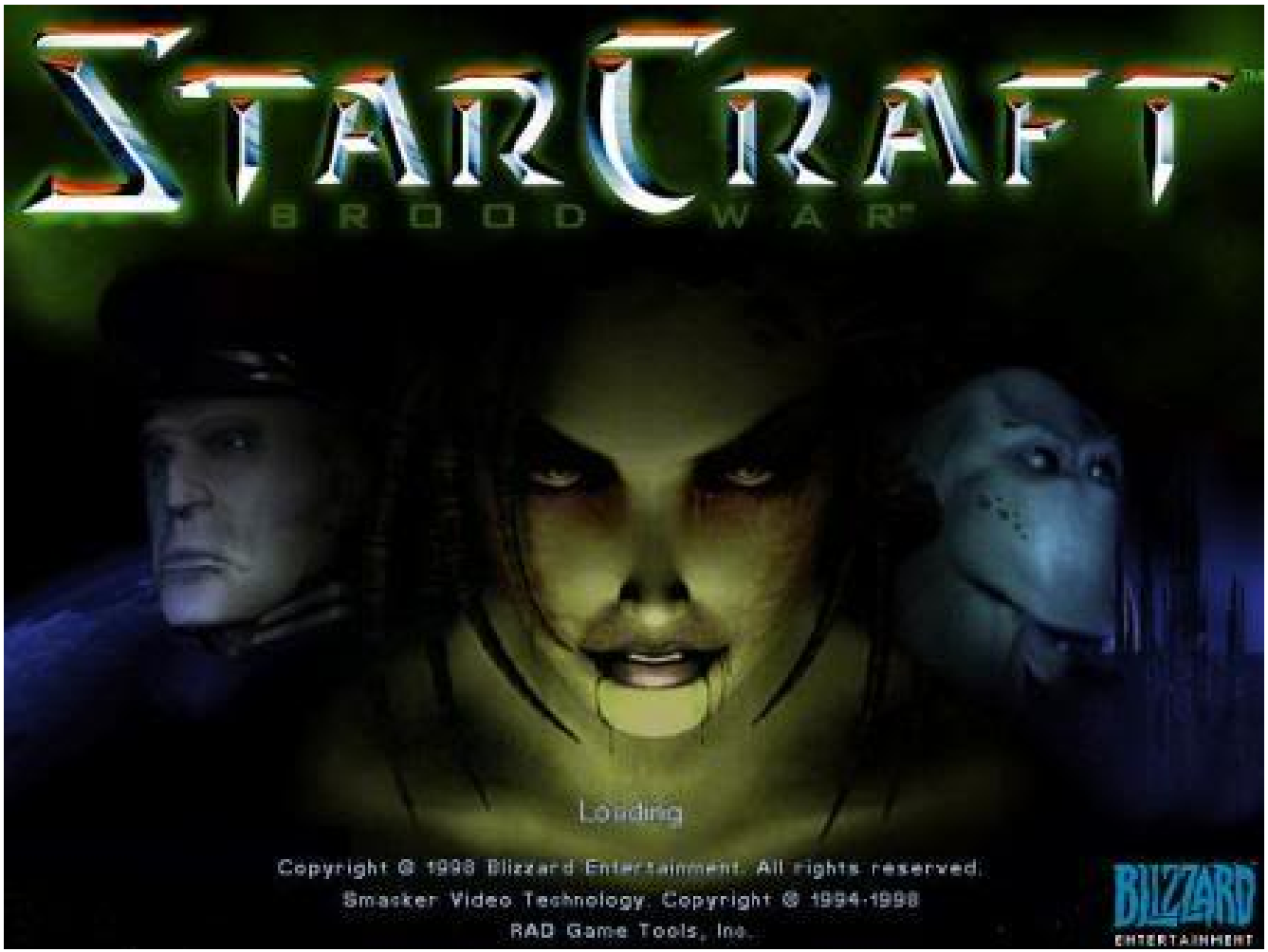
University of Grenoble
LPPA @ Collège de France (Paris)
E-Motion team @ INRIA (Grenoble)

September 5, 2011

- Introduction
  - StarCraft
  - Our Approach
- Part 1: Micro-Management
  - Problem
  - Model
  - Results
- Part 2: Enemy Strategy Prediction
  - Problem
  - Model
  - Results
- Conclusion
  - Summing-Up
  - Future Work

Starcraft: Broodwar

Starcraft (January 1998) + Broodwar (exp., November 1998)



Pro gaming and competitions

eSports, sponsorship, tournaments' dotations

WCG
WORLD CYBER GAMES

WORLD CYBER GAMES

The World Cyber Games is the world's first "Cyber Game Festival", designed to build a healthy cyber culture.
The best gamers around the world gather into different cities to share the excitement and fun of the game tournaments.

ESWC
ELECTRONIC SPORTS WORLD CUP

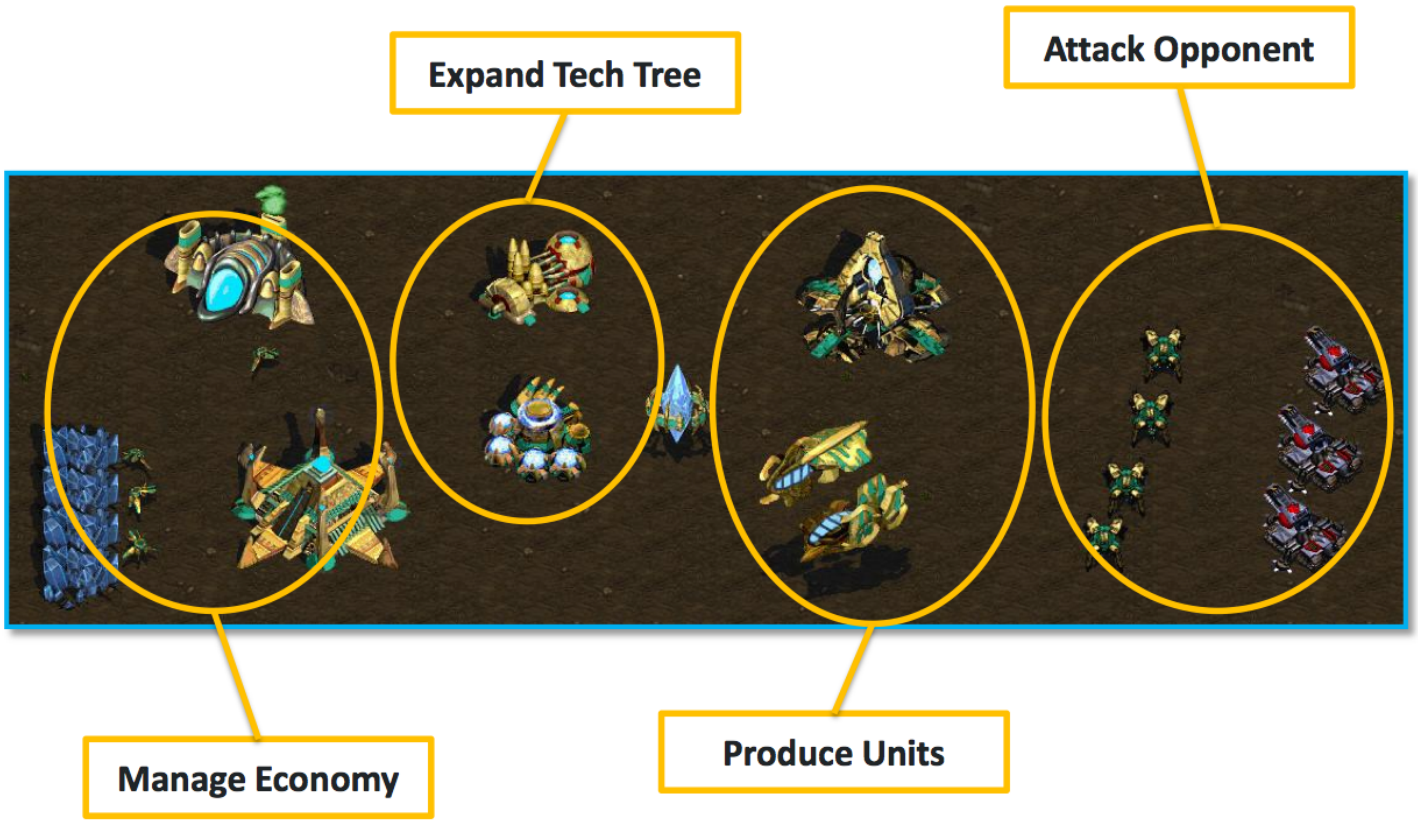
Gosu Gamers

teamliquid
STARCRAFT PROGAMING NEWS • COMMUNITY • TEAM

Starcraft in numbers

- **12** years of competitive play
- **200** to **300** actions per minute amongst pro gamers
- **10** millions licenses sold (4.5 in South Korea)
- **160** BPM: reached rates of pro gamers hearts
- **4.5+** millions licenses sold for Starcraft II

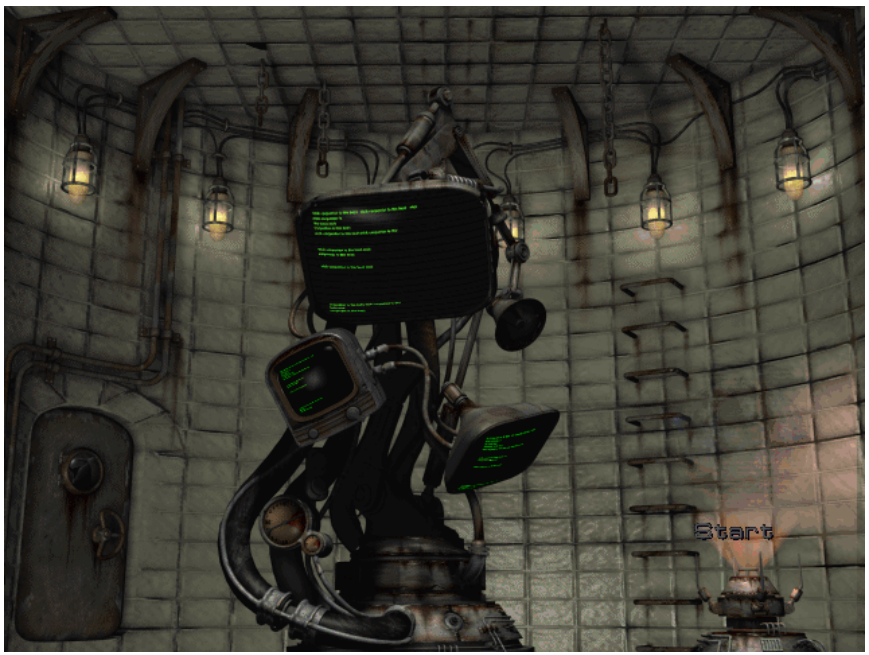
Real-Time Strategy Game



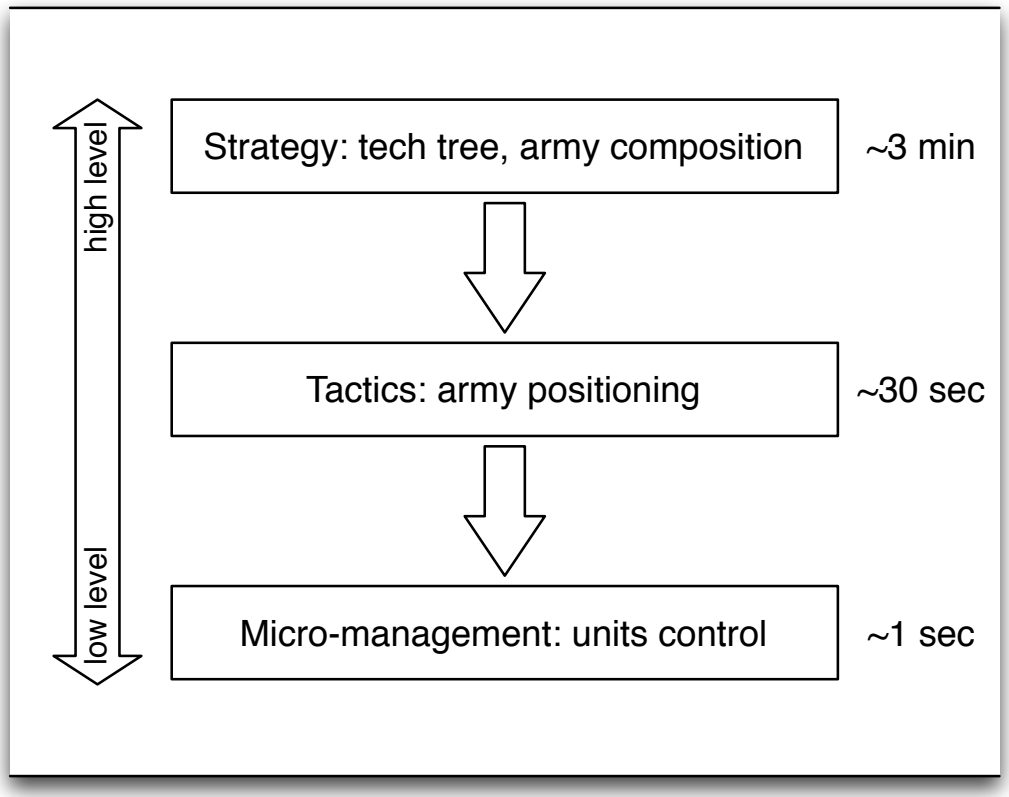
("shamelessly stolen with permission" from Ben Weber, UCSC)
The models and approaches presented here are also **valid** in Total Annihilation, Age of Empires and Warcraft 2.

Interest for RTS games

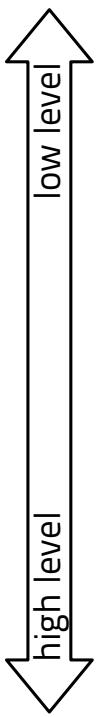
- Chess / Go / Rock-paper-scissors
- Real-time (1/24th second per micro-turn)
- Machine learning ready (supervised, unsupervised, reinforcement)
- AI competitions



Problems to tackle



Transmute incompleteness into uncertainty



Incompleteness $\Longrightarrow$ Uncertainty

- Many low level moves achieving the same high level goal
- Fog of war (limited sight)
- Partial knowledge of opponent's force (size and composition)

- Considering the units as individual Bayesian robots
- Seen units (viewed units filter)
- Probabilistic inference, machine learning from *replays*

A Bayesian program structure

$$BP \begin{cases} Desc. \begin{cases} Spec.(\pi) \begin{cases} Variables \\ Decomposition \\ Forms\ (Parametric\ or\ Program) \end{cases} \\ Identification\ (based\ on\ \delta) \end{cases} \\ Question \end{cases}$$

$$P(Searched|Known)$$

$$= \frac{\sum_{Free} P(Searched,\ Free,\ Known)}{P(Known)}$$

$$= \frac{1}{Z} \times \sum_{Free} P(Searched,\ Free,\ Known)$$

A Bayesian program example: the Kalman filter

Description

Specification

Variables:

$S^0, \ldots S^t, O^0, \ldots, O^t$

Decomposition:

$P(S^0, \ldots, S^t, O^0, \ldots O^t)$

$= P(S^0).P(O^0|S^0).\Pi_{i=1}^{t}[P(S^i|S^{i-1}).P(O^i|S^i)]$

Parametric forms:

$P(S^0) = G(S^0, \mu, \sigma)$

$P(S^i|S^{i-1}) = G(S^i, A.S^{i-1}, Q)$

$P(O^i|S^i) = G(O^i, H.S^i, R)$

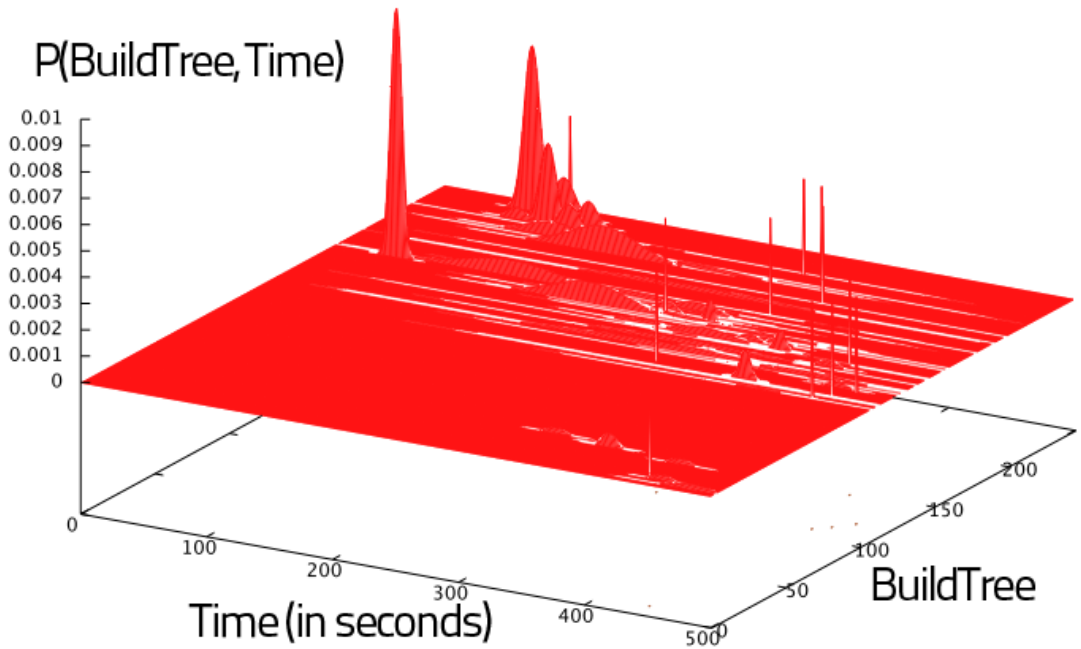
Identification — Learning from instances

Question

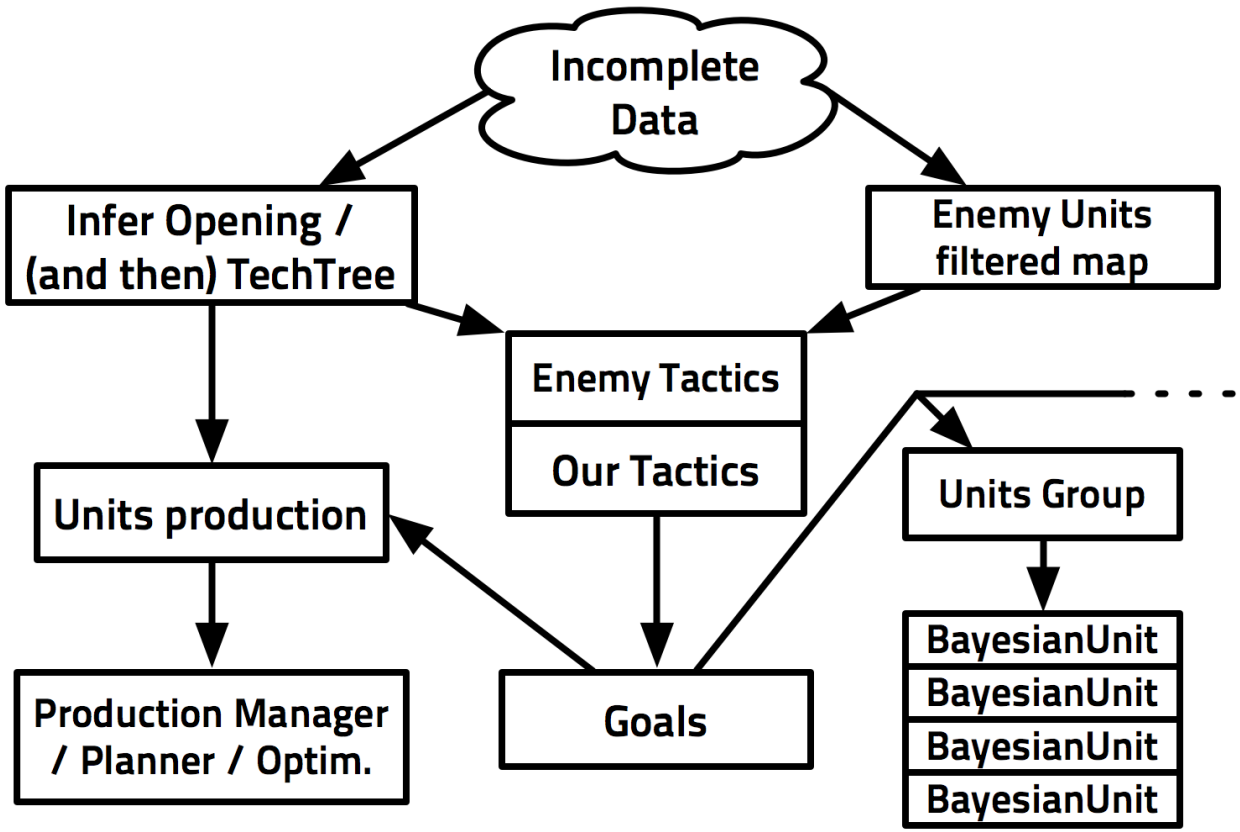
$P(S^t|O^0, \ldots, O^t)$

Machine learnings

- from replays (parameters of predictive models)
- reinforcement (exploration of parameters space for the Bayesian robots)
- online (adapt to particular opponent)



Model overview



Not a perfect (nor what-we-want-in-the-end) model, but the actual, implemented, bot model.

Part 1

Micro-management

Micro-management in RTS games

Micro-management is the art of maximizing the efficiency of your units:

- Focus-fire enemy units to reduce their fire power,
- Move away damaged units (“dancing”),
- Allow damaged units to flee (collisions), best placements w.r.t. wanted/best target possible.

Open question: for humans, another objective is to rank the actions by importance (APM limit).

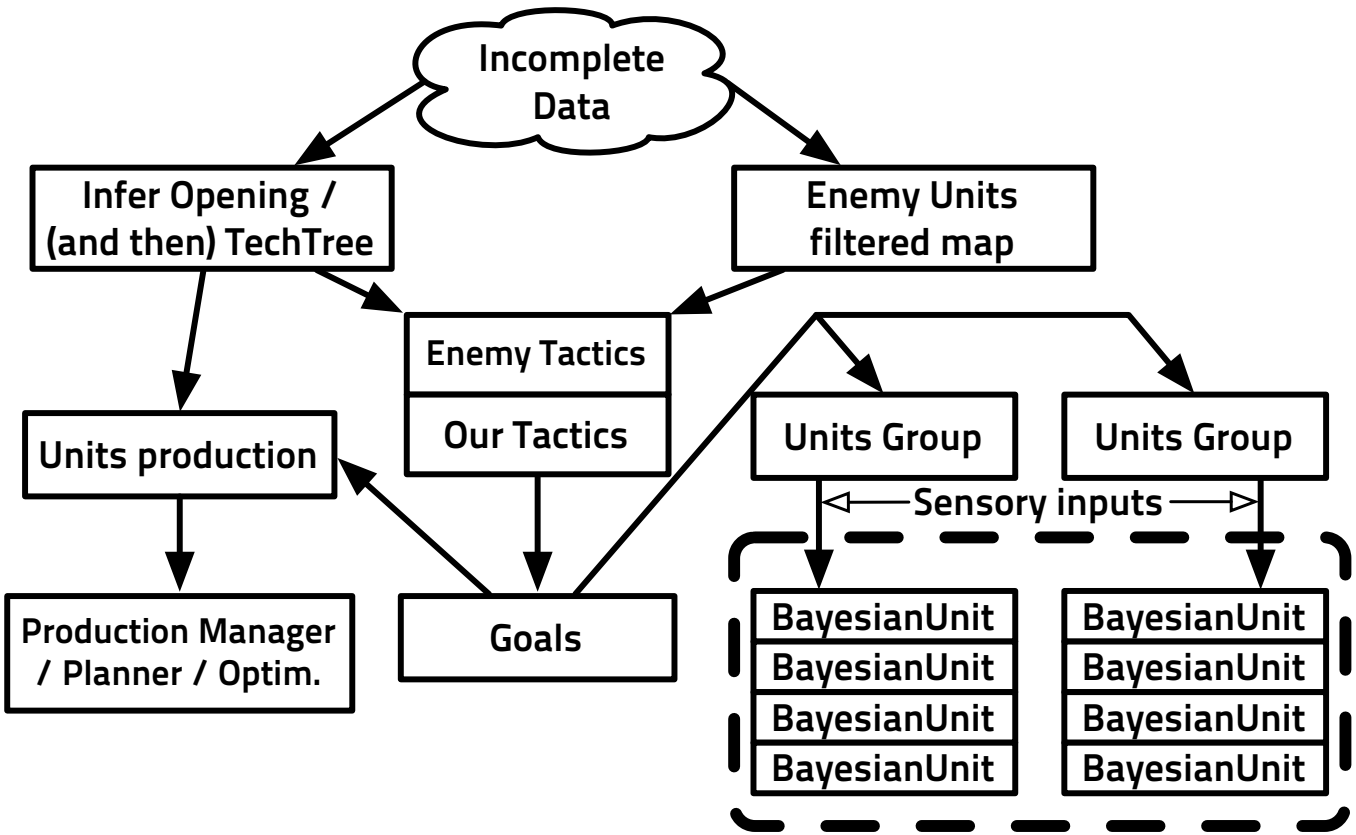
Serious Stuff (real-time Chess)

24 game simulation FPS, fine grained discrete world (almost continuous), **branching factor** out of control, **uncertainty** about opponent's movements.

$\Rightarrow$ "No" computable optimal solution.

Our take: **units as independant Bayesian robots**.

Where are we?



Here

The Bayesian unit (1)

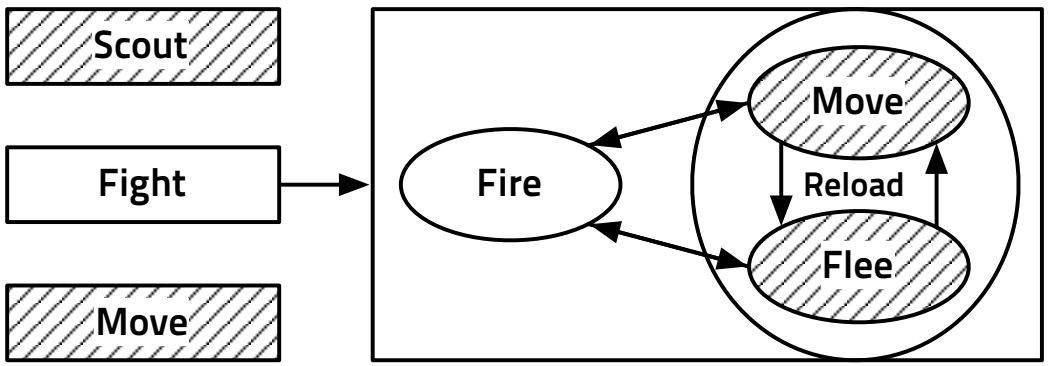
The workers are *not* controlled as (Bayesian) sensory motor systems as their tasks are extremely simple.

Modal unit, can be simplified as HFSM:

- Scout
- Move (small / large group)
- In position
- Fight:
  - Attack
  - Fight-move
  - Flee

} When not attacking

The Bayesian unit (2) [Firing]



Focus-firing heuristic (simplified): Fire on units that do the most damages, have the less hit points, and take the most damages from their attack type.

Inputs to our unit:

- wanted/best target: best target w.r.t heuristic
- immediate target: best target **in firing range**

The Bayesian unit (3) [Moving/Fleeing]

Mode dependant influences, as:

- “statically” occupied tiles
- “dynamically” occupied tiles
- height
- damage map (+ gradient)
- pathfinder
- ...

Fight-move example (1)

Variables:

- Dir n values with $n = \#\{\text{possible directions}\}$,
- $Dir_{i \in \{\text{possible directions}\}} \in \{T, F\}$,
- $Obj_i \in \{T, F\}$, direction of the objective (quantified)
- $Dam_i \in \{no, low, med, high\}$, subjective potential field,
- $A_i \& E_i \in \{none, small, big\}$, allied & enemy units presence,
- $Occ_i \in \{no, terrain, building\}$, "static" occupation

Decomposition:

$$
\begin{aligned}
& P(Dir, Dir_{1:n}, Obj_{1:n}, Dam_{1:n} Rep_{1:n}, Occ_{1:n}) \\
= \Pi_{i=1}^{n} \quad & P(Dir_i | Dir) \;//\; Dir = i \Rightarrow 1, Dir \neq i \Rightarrow 0 \\
& .P(Obj_i | Dir_i) \\
& .P(Dam_i | Dir_i) \\
& .P(A_i | Dir_i).P(E_i | Dir_i \\
& .P(Occ_i | Dir_i)
\end{aligned}
$$

Fight-move example (2)

Parameters:

They are hand-specified for the moment but should be learned through task-specific maps and maximization (EA/GP, RL).

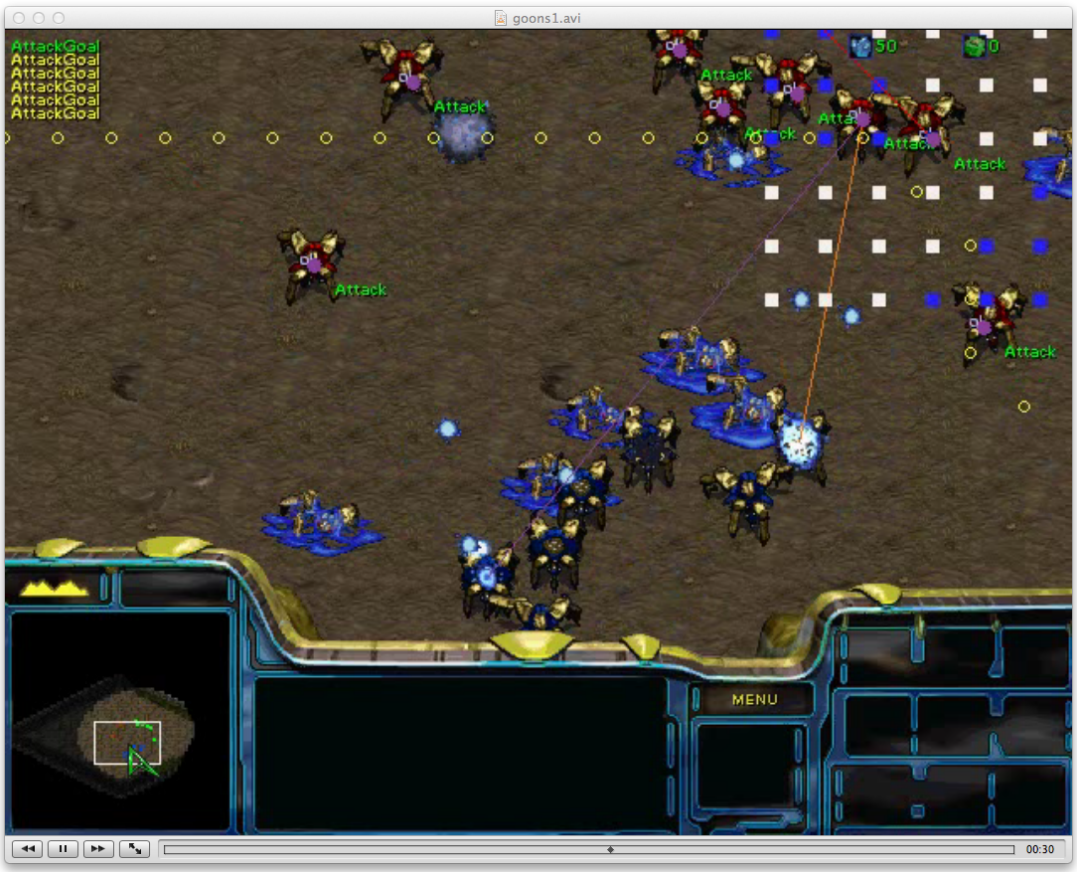
Questions:

Our pathfinder gives us $Obj_{1:n}$, the damage map $Dam_{1:n}$, the size (or number) of other units in tile/direction i $A/E_{1:n}$, etc.

$$P(Dir|Obj_{1:n}, Dam_{1:n}, A_{1:n}, E_{1:n}, Occ_{1:n})$$

Fight-move example (3)

Video: a fight in AIIDE micro-tournament setup 2 (decisions of movements taken by sampling on *Dir*):



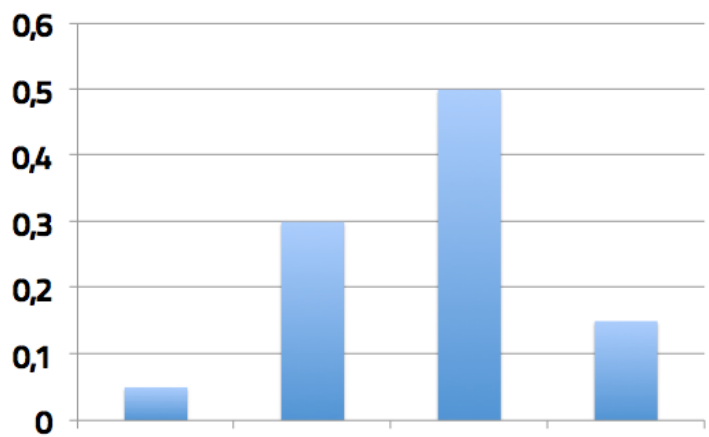
Bayesian flocking (simplified)

Variables:

- $Dir_{i \in \{\text{possible directions}\}} \in \{T, F\}$,
- $Obj_i \in \{T, F\}$,
- $Att_i \in \{too\ close, close, far, too\ far\}$, closeness to allied units for the ith direction (for instance),

Parameters:

$P(Att_i|Dir_i)$ can be learned with maze-like maps and on the objective to minimize the time-to-completion (will find the optimal flocking distance/attraction-repulsion):



Repulsion applied “in position”

(2 videos which can be found on youtube
`http://www.youtube.com/snippyhollow#p/a/u/1/sMyF_PlDqFo`
`http://www.youtube.com/snippyhollow#p/a/u/2/mvv9kUntLHU`)
Variables: Dir_i, Obj_i, Rep_i, but with different parameters for $P(Rep_i|Dir_i)$ (can be learned also), and a different objective: the wanted unit position.

AIIDE 2010 Tournament 1

- 7 participants
- Could not take part (technical problem)
- FreSC (France, Epita team) won
- Played against FreSC: draw! (0-3, 3-0, draw (host wins))

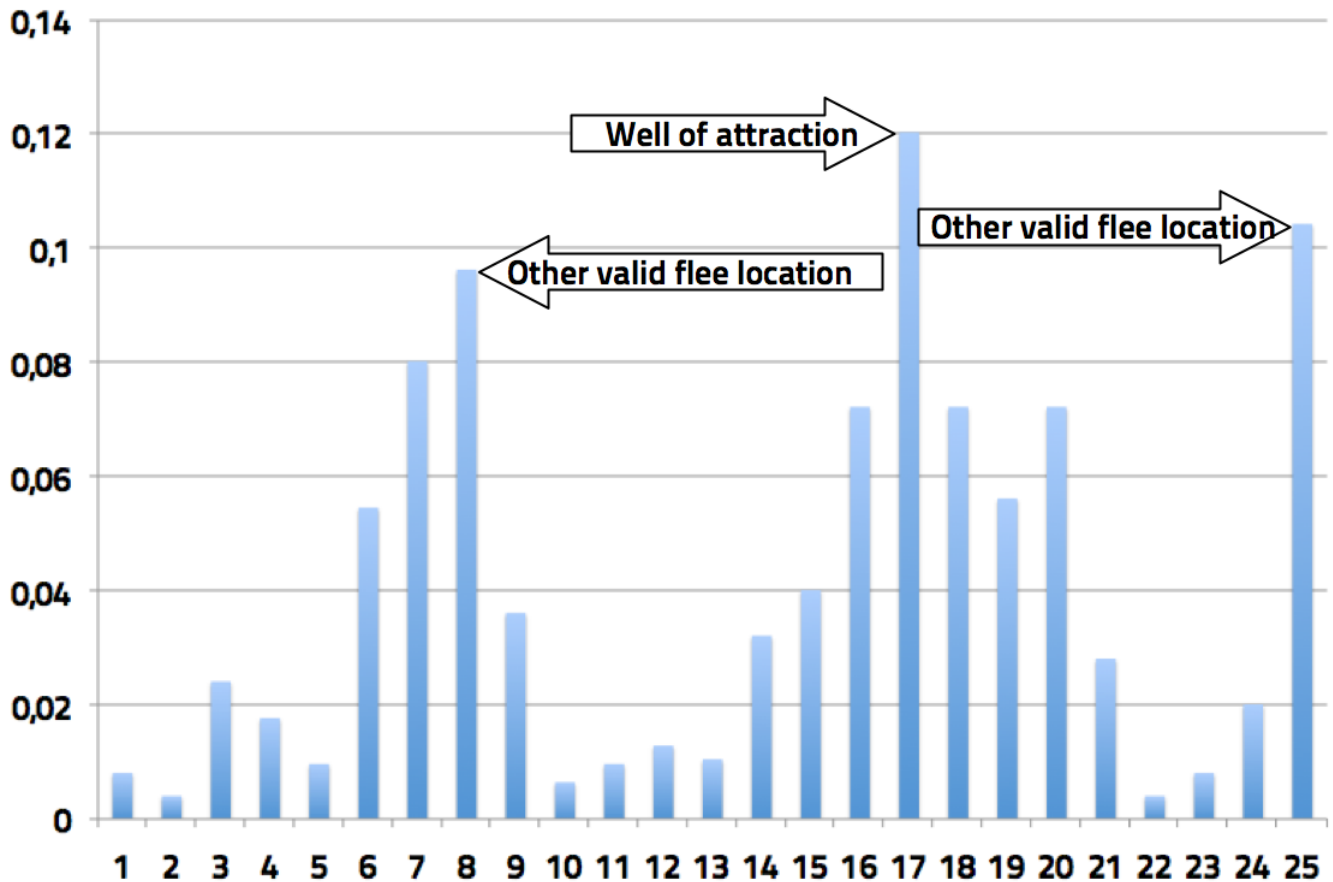
Benchmarking the movements efficiency

We designed/implemented a bot without the movements advantage to highlight the increase in value of our model: a *target selection heuristic only* bot (focus-firing bot).

Big Populations Blunder: Collisions

Simplest solution

Sample in the *Dir* distribution!



(Note that our AIIDE 2010 bot was using the “pick best” policy.)

Recap. Table

12 units \ 36 units	OAI	HOAI	BAIPB	BAIS
OAI	(50%)	64%	9%	3%
HOAI	59%	(50%)	11%	6%
BAIPB	93%	97%	(50%)	3%
BAIS	93%	95%	76%	(50%)

Win ratios over at least 200 battles of Original AI, Heuristic Only AI, Bayesian AI Pick Best, Bayesian AI Sampling in two mirror setups: 12 and 36 ranged units. Read line vs column win %.

A word on parameters

Parameters can be hand specified, so we can provide **game designers** with a slider on a λ or μ parameter controlling the distribution on the model's probability tables. For instance, $P(Dmg_i|Dir_i = T)$ controls the risk-taking/bravery/temerity of the unit. Game designers can toy around with behaviours this way.

In the framework of a **competition**, we can learn these parameters to maximize the objective. Being it a micro-management only or a complete game (objectives differ a little: the higher tactic level can decide to sacrifice a group on an objective).

Next?

A lot of possible improvements/follow-ups:

- Other sensory inputs (heights/tactical positions...),
- Learning the tables (EA/GP/RL for big combinations) w.r.t. situations,
- Either a case-based approach for situation recognition or situations recognition tables fusion,
- Group controllability vs unit autonomy ($\approx$ unit recklessness/sacrifice).

Part 2

Strategy Prediction

Definition

Infer what the enemy **opening**[1] is from partial observations (because of the *fog of war*) to be able to deal with it, or counter it if we can.

(Another problem is then to dynamically adapt our own opening/strategy.)

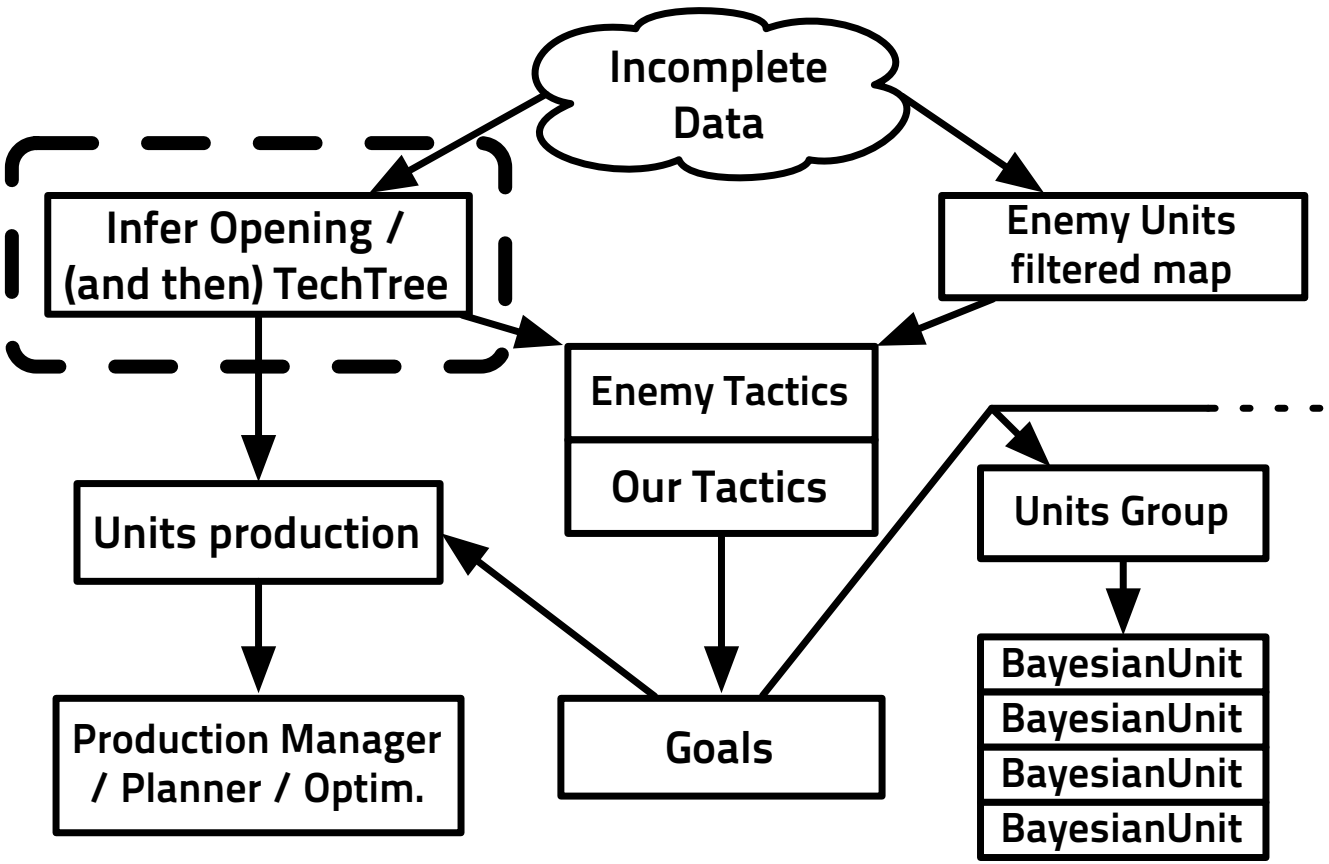
[1] opening = first strategy = first units + first tactical move (as in Chess), we will reserve the term strategy for *army composition* + *long term tactical goals*)

Examples of openings (cheeses)

All-in fast dark templars:
Produce 2 dark templars (permanently cloaked unit type) as fast as possible to destroy the enemy while he can not detect them. *Attempt to finish the game with a very specific (and* ***weak when countered****) unit deep in the tech path.*

All-in 2 gates zealots rush:
Produce only zealots (lowest tech Protoss military unit), stream them once a critical attack mass (6+) is reached. *Attempt to finish the game before the opponent's economy or technological ROI kicked in.*

Where are we?



Here

A word on the meta-game

Openings prior probabilities are influenced by maps and previous games (meta-game) in the same match and/or against the same opponent (pros tourneys: BO5+). We will not consider it for the rest of this work. **This is a huge mistake**[2], as it is central to StarCraft gameplay balance, but we play by the rules of current bots tournaments (BO1).

[2] it is very easy to adapt in our model: just put/change a prior

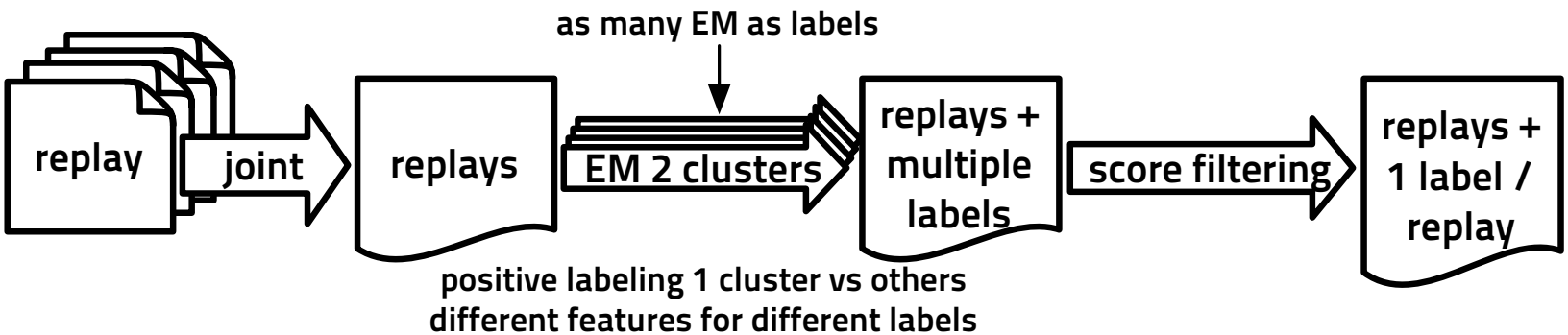
Replays

Record all the actions of the player so that the game can be deterministically re-simulated (random generators seeds are serialized).

Supervised learning model:

- Need for the replays to be annotated with openings.
- Used Ben Weber (UCSC) dataset (annotated with tech tree deployment order / **rules**) for comparisons purposes: 9316 games, between 500 and 1300 per match-up.
- Can use other (more) replays (as we can automatically annotated them).

Replays openings labeling



Semi-supervised labeling: manually extracted features + clustering + annotation heuristic (“earliest happening” cluster is labeled positively).

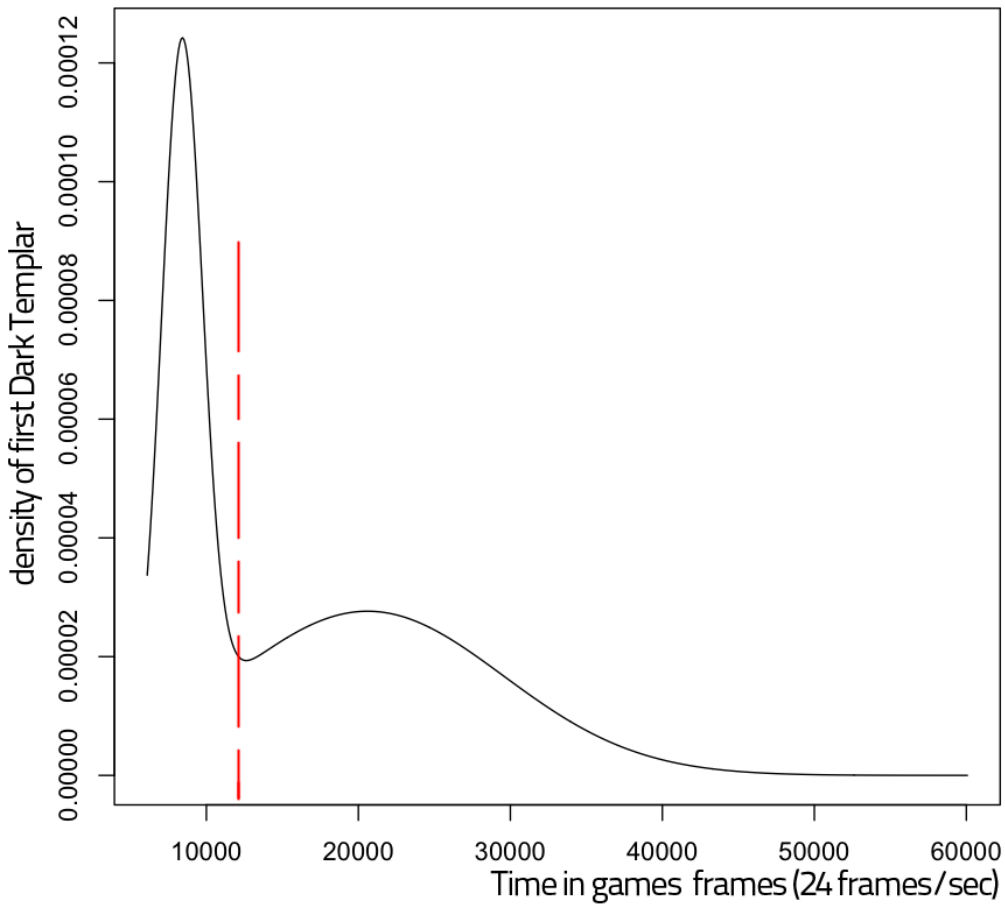
Features selection

Features selected for each label (opening value) that we want to put. From experts knowledge.
For instance:

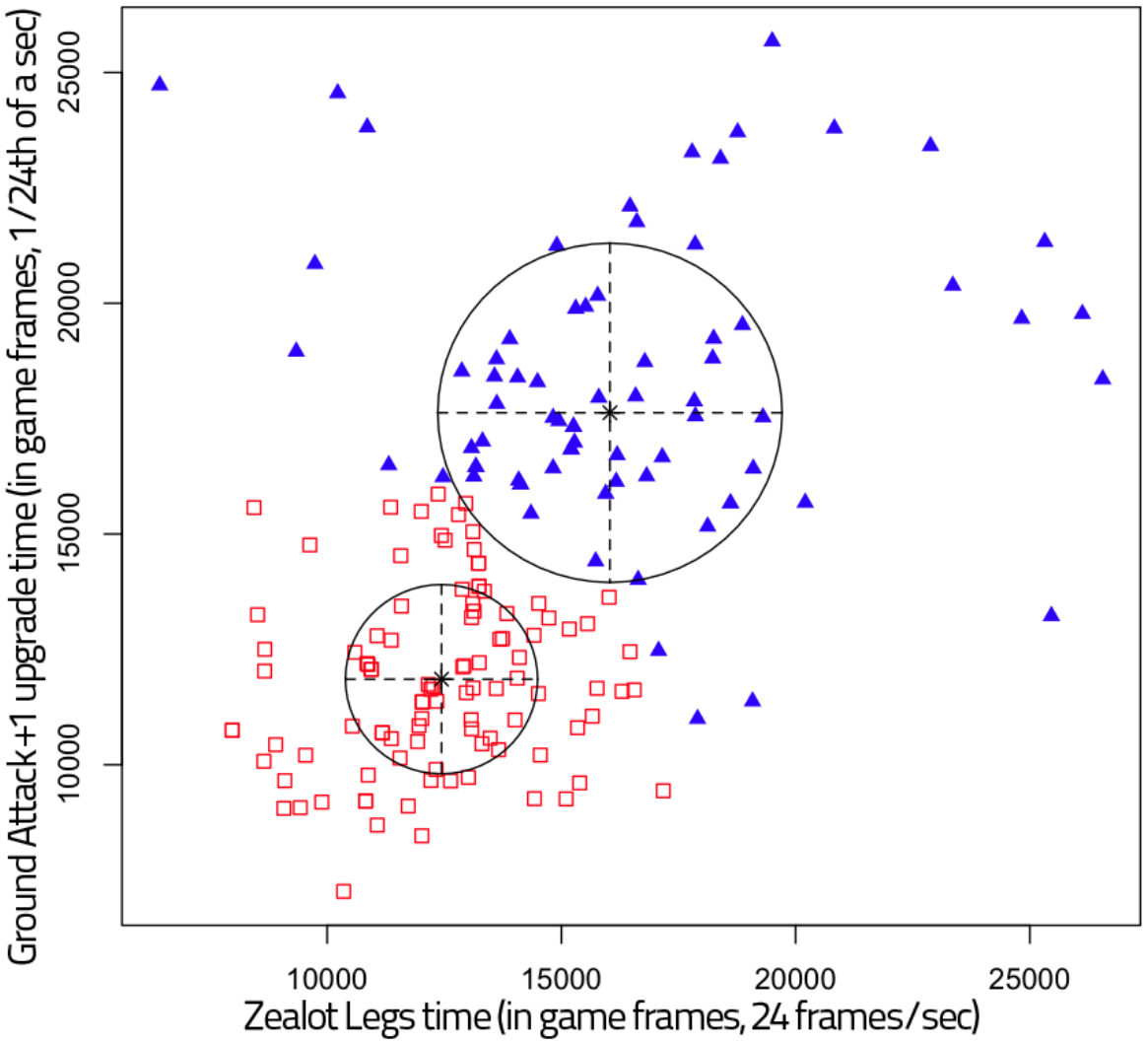
- what is important for the “fast dark templars” opening is the time of the first production of a dark templar.
- what is important for the “2 gates rush” opening are the time of constructions of the first **and** second gate, and the time of production of the first zealot.

Clustering (1), PvT Fast DT opening

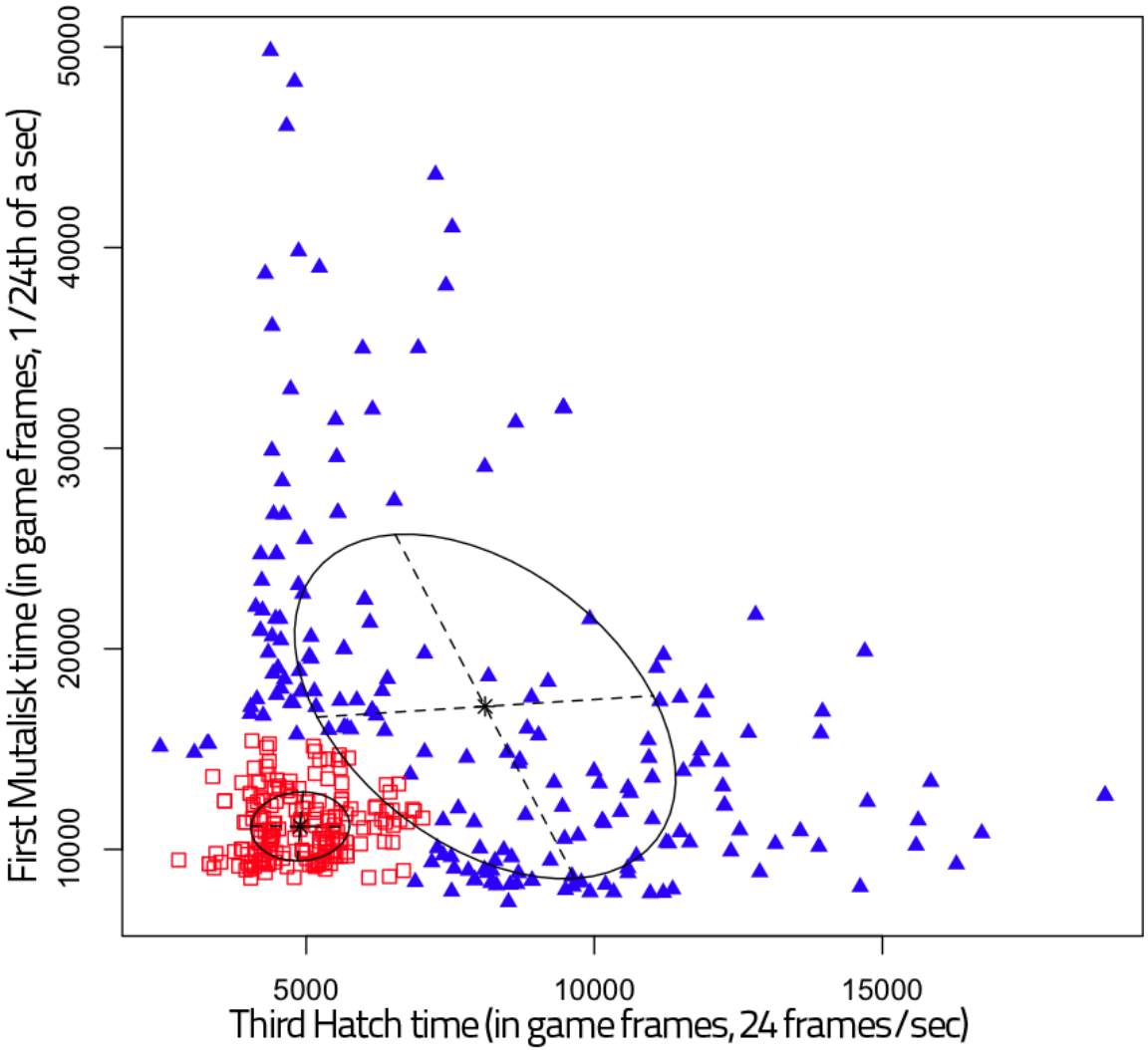
Tried many techniques, used Mclust (R) full EM (best results):



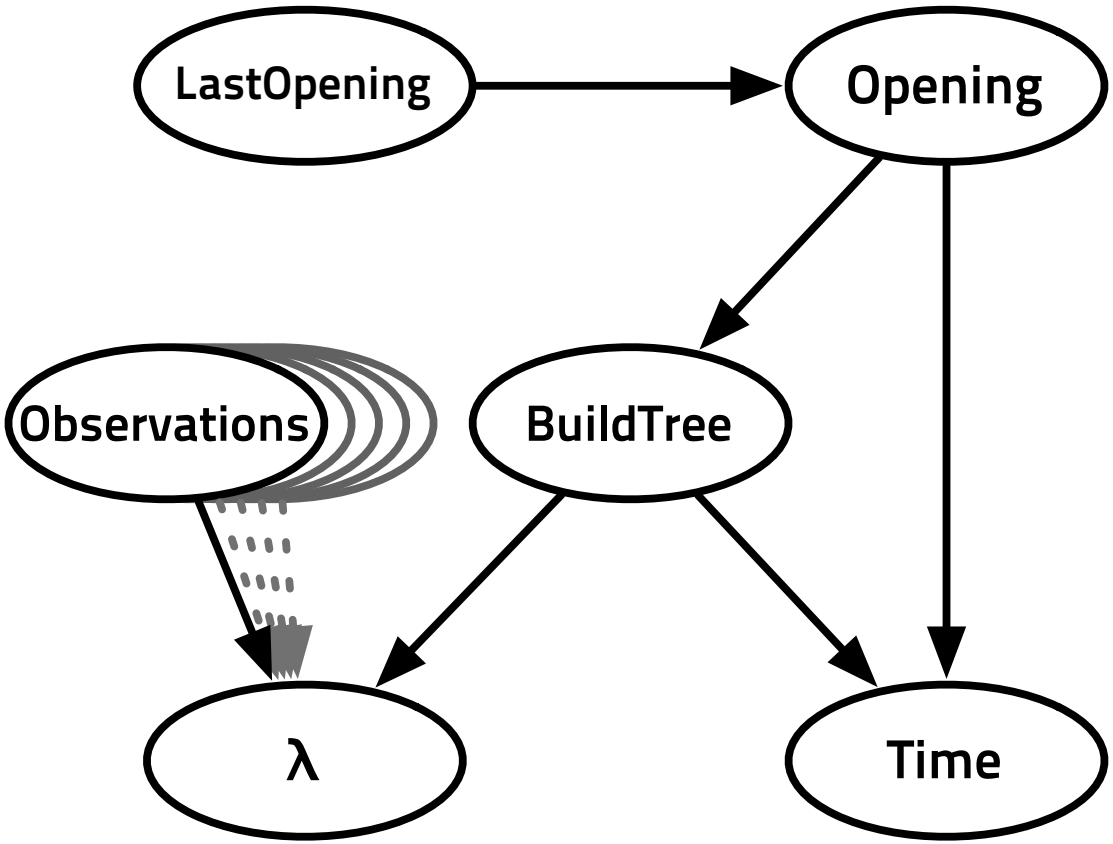
Clustering (2), PvP Speed zealots opening



Clustering (3), ZvP Fast mutas opening



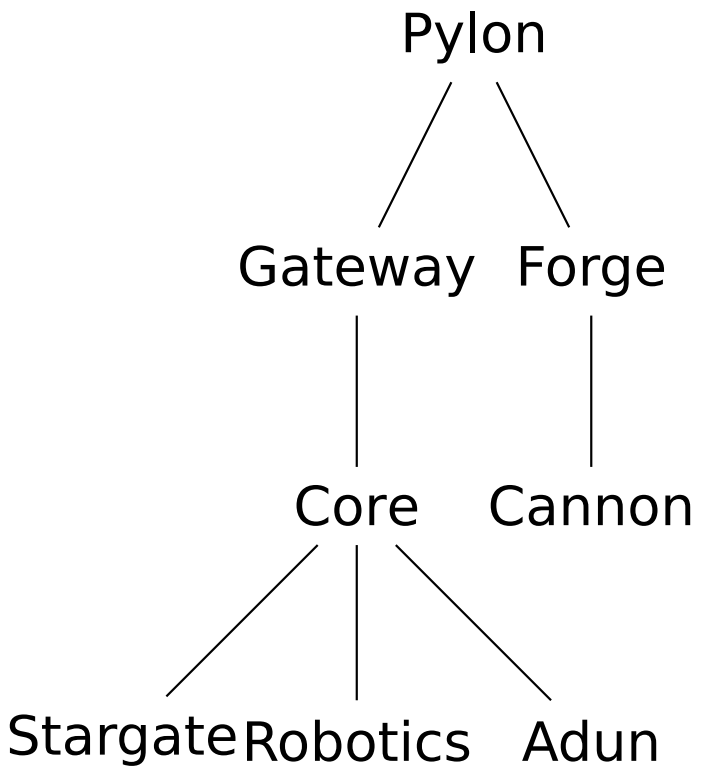
Bayesian Model



Variables

- $BuildTree \in [\emptyset, building_1, building_2, building_1 \wedge building_2, buildtrees, \dots]$
- N *Observations*: $O_{i \in [\![1 \dots N]\!]} \in \{0, 1\}$, O_k is 1 (*true*)
- *Opening*: $Op^t \in [opening_1 \dots opening_M]$
- *LastOpening*: $Op^{t-1} \in [opening_1 \dots opening_M]$
- $\lambda \in \{0, 1\}$: coherence variable (restraining *BuildTree* to possible values with regard to $O_{[\![1 \dots N]\!]}$)
- *Time*: $T \in [\![1 \dots P]\!]$

BuildTree variable by example



$$BuildTree \in \{\emptyset, \{Pylon\}, \{Pylon, Gateway\}, \{Pylon, Forge\},$$
$$\{Pylon, Gateway, Forge\}, \{Pylon, Gateway, Core\}, \dots\}$$

Decomposition

$$
\begin{aligned}
& P(T, BuildTree, O_1 \dots O_N, Op^t, Op^{t-1}, \lambda) \\
= \quad & P(Op^t | Op^{t-1}) \\
& P(Op^{t-1}) \\
& P(BuildTree | Op^t) \\
& P(O_{[\![1 \dots N]\!]}) \\
& P(\lambda | BuildTree, O_{[\![1 \dots N]\!]}) \\
& P(T | BuildTree, Op^t)
\end{aligned}
$$

Forms

- $P(Op^t|Op^{t-1})$, a filter so that the previous inference impacts the current one (functional Dirac)
- $P(BuildTree|Op^t)$, histogram learned from replays
- $P(\lambda|BuildTree, O_{[\![1...N]\!]})$ restricts *BuildTree* values to the ones that can co-exist with the observations
- $P(T|BuildTree, Op^t)$ are discretized normal distributions. There is one bell shape per $(opening, buildTree)$ couple. The parameters of these discrete Gaussian distributions are learned from the labeled replays.

A note on identification/learning

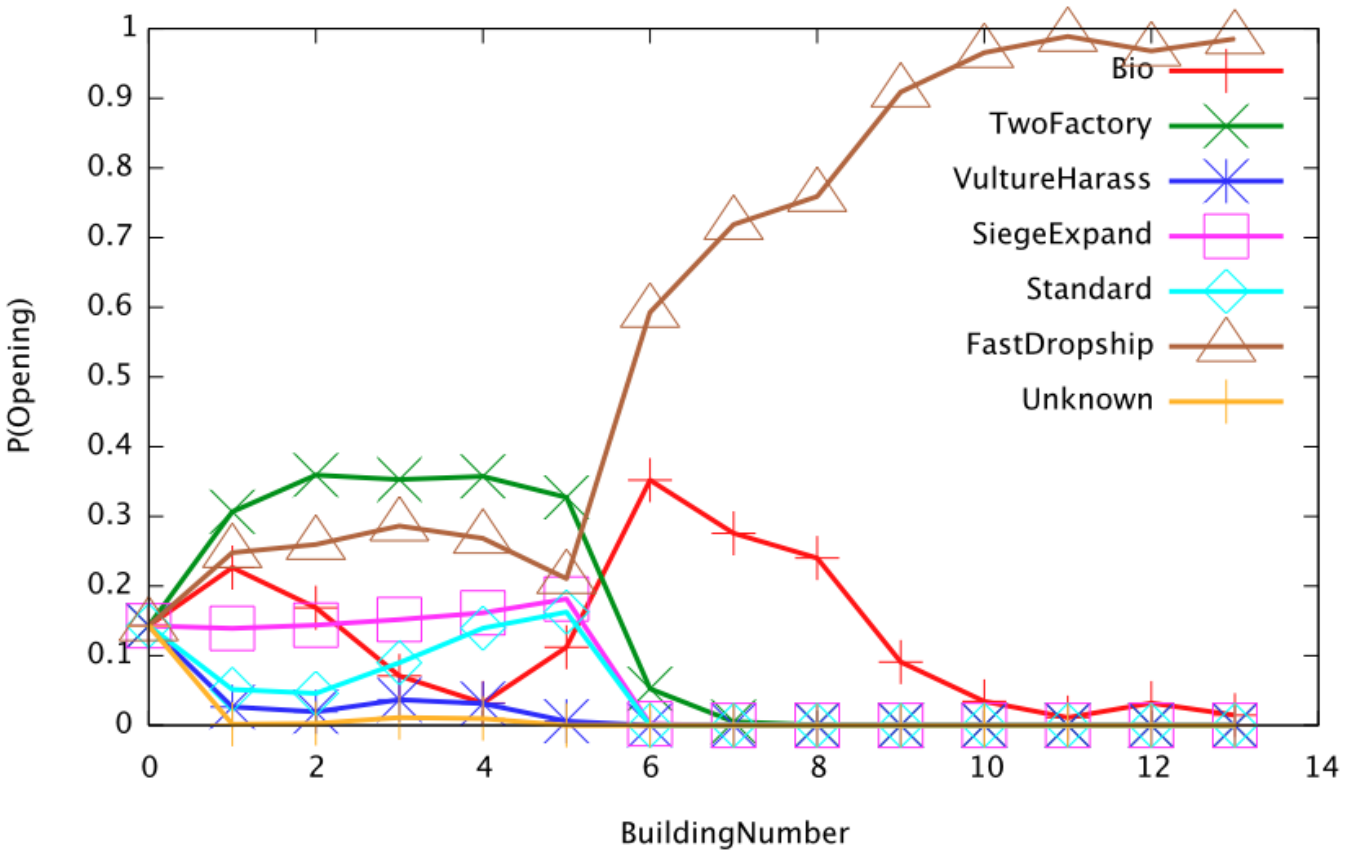
- Learning of the $P(T|BuildTree, Op^t)$ bell shapes parameters takes into account the uncertainty of the couples $(buildTree, opening)$ for which we have few observations by starting with a high σ^2.
- Learning on human replays for bots opening recognition does not work well. We had to impose a large minimal σ^2 (more robustness at the detriment of precision).

Question

$$P(Op|T = t, O_{[\![1...N]\!]} = o_{[\![1...N]\!]}, \lambda = 1)$$
$$\propto P(Op).P(o_{[\![1...N]\!]})$$
$$\times \sum_{BuildTree} P(\lambda|BuildTree, o_{[\![1...N]\!]})$$
$$.P(BuildTree|Op).P(t|BuildTree, Op)$$

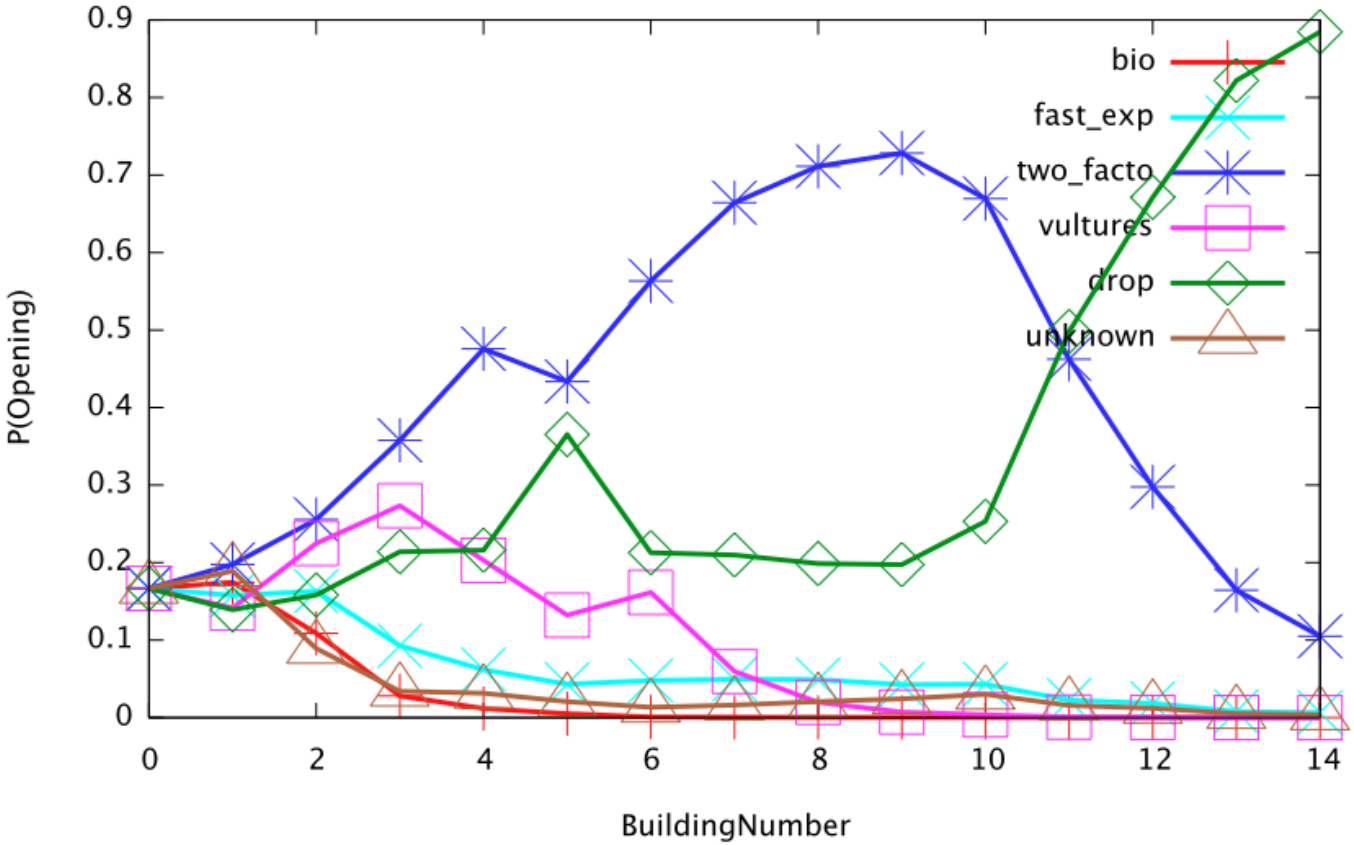
Note: $P(BuildTree|Opening, Time)$
would use the plan recognition model as a planning model.

Evolution of predictions (1)



Ben’s labels

Evolution of predictions (2)



Our labels

Low CPU and memory footprint

Race	# Games	Learning time	Inference μ	Inference σ^2
T	1036	0.197844	0.0360234	0.00892601
T	567	0.110019	0.030129	0.00738386
P	1021	0.13513	0.0164457	0.00370478
P	542	0.056275	0.00940027	0.00188217
Z	1028	0.143851	0.0150968	0.00334057
Z	896	0.089014	0.00796715	0.00123551

In game prediction

(Video: watch the right of the screen.)
`http://www.youtube.com/watch?v=7ycEkK54lTg`

Used it in BroodwarBotQ, free software (BSD 3-clauses):
`http://github.com/SnippyHolloW/BroodwarBotQ`

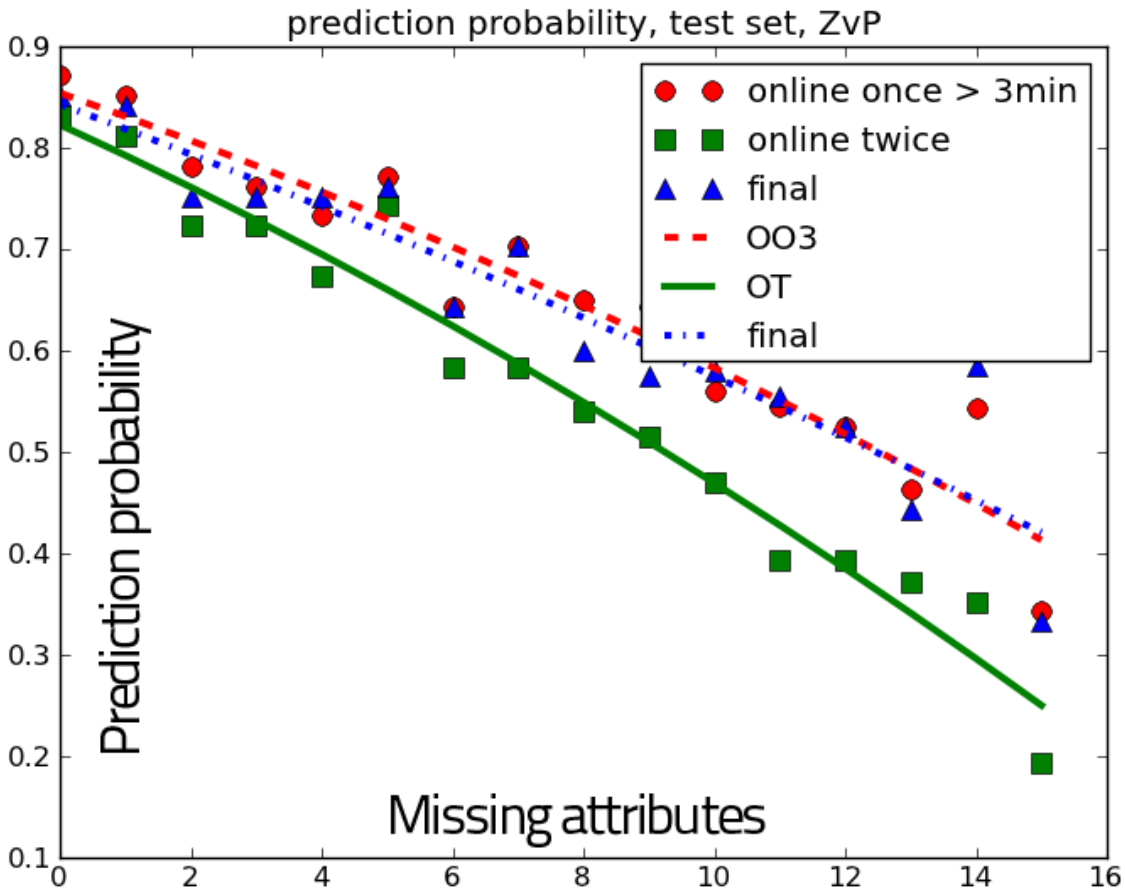
Evaluation metrics

- the *final* prediction, opening that is predicted at the end of the test.
- the *online twice* (OT), counts the openings that have emerged as most probable twice during a test (not due to noise),
- the *online once* > 3 (OO3), counts the openings that have emerged as most probable openings after 3 minutes (meaningful information).

Recap. performance table

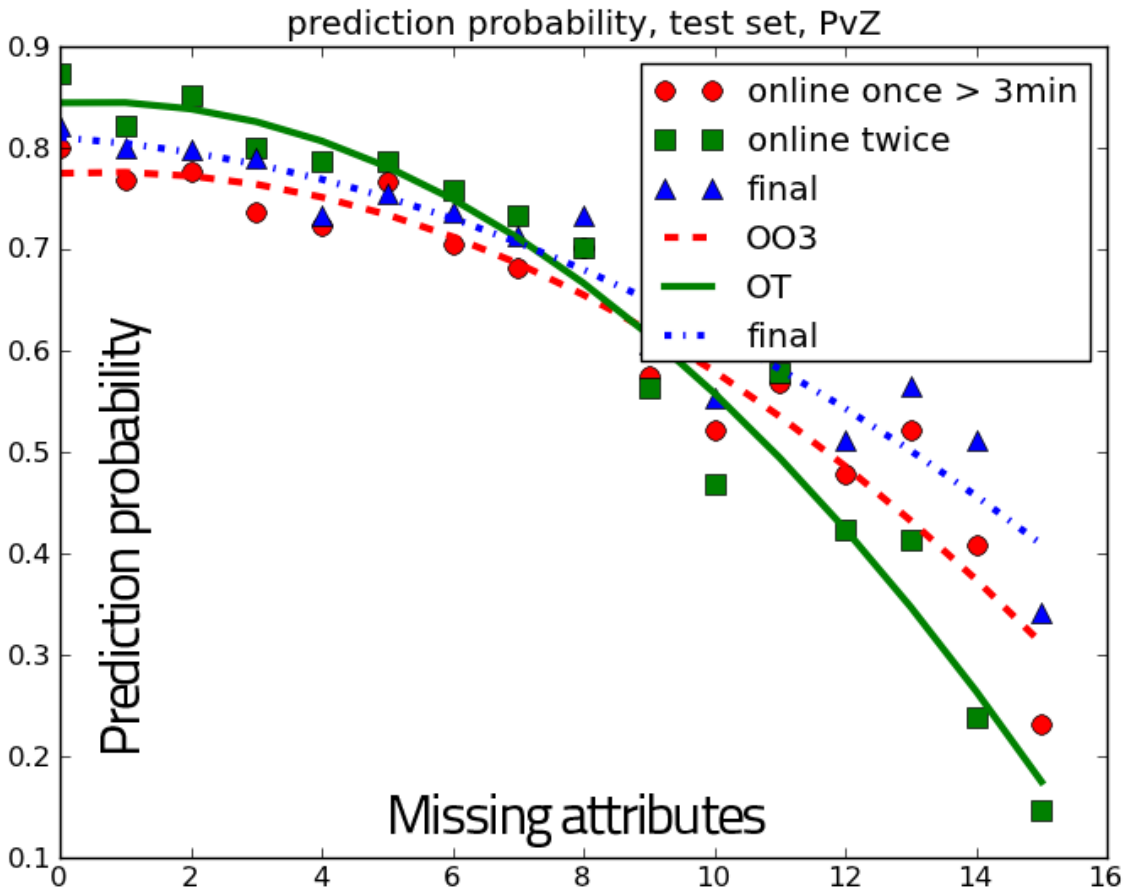
	Weber and Mateas' labels					Our labels				
	5 minutes		10 minutes			5 minutes		10 minutes		
match-up	final	OO3	final	OT	OO3	final	OO3	final	OT	OO3
PvP	0.65	0.59	0.69	0.69	0.71	0.78	0.68	0.83	0.83	0.83
PvT	0.75	0.71	0.78	0.86	0.83	0.62	0.69	0.62	0.73	0.72
PvZ	0.73	0.66	0.8	0.86	0.8	0.61	0.62	0.66	0.66	0.69
TvP	0.69	0.76	0.6	0.75	0.77	0.50	0.54	0.5	0.6	0.69
TvT	0.57	0.65	0.5	0.55	0.62	0.72	0.77	0.68	0.89	0.84
TvZ	0.84	0.81	0.88	0.91	0.93	0.71	0.77	0.72	0.88	0.86
ZvP	0.63	0.64	0.87	0.82	0.89	0.39	0.52	0.35	0.6	0.57
ZvT	0.59	0.59	0.68	0.69	0.72	0.54	0.61	0.52	0.67	0.62
ZvZ	0.69	0.67	0.73	0.74	0.77	0.83	0.85	0.81	0.89	0.94
overall	0.68	0.68	0.73	0.76	0.78	0.63	0.67	0.63	0.75	0.75

Tolerance to noise (1)



Zerg (versus Protoss) opening recognition with increasing noise (15 missing attributes $\leftrightarrow$ 93.75% mission information).

Tolerance to noise (2)



Protoss (versus Zerg) opening recognition with increasing noise (15 missing attributes $\leftrightarrow$ 88.23% mission information).

Comparing results with existing works

Compared to previous work by Ben Weber [CIG 2009]:

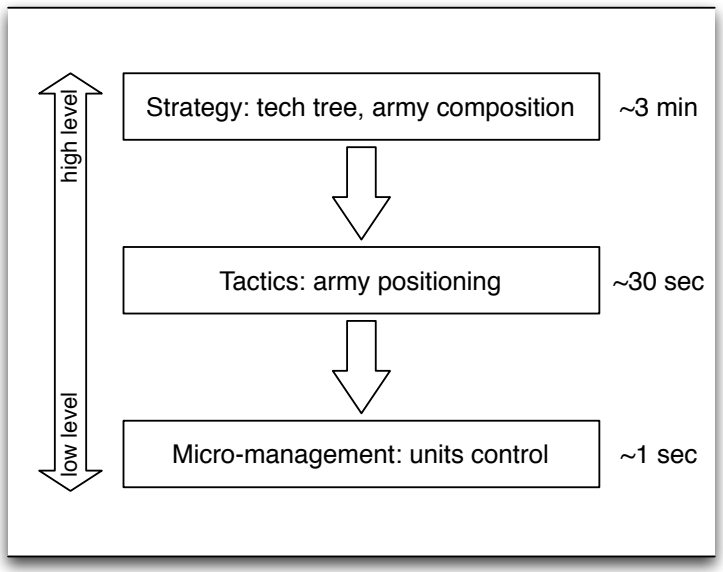
- Works with partial information (fog of war),
- Resists quite well to noise,
- Gives a distribution, not just a decision (that's how high level human player think).

Kudos to Ben, who was a nice enough pioneer to distribute his dataset.

Limitations

- Dependant on the labeling: while I think ours is better than Ben's, it's not *systematically* the case,
- Can not deal with multiple-labels (*Unknown* label contains some unexploited information),
- Should use some other (sometimes more advanced, not necessarily more numerous) features.

Things that work



- Micro-management “in general” is quite efficient and *robust* (could be more optimized and more controllable),
- Opening/Strategy recognition/prediction gives *good enough* results online.

Possible Improvements

Direct possible improvements:

- Micro-management **optimization** for **particular situations** through **learning** (see Part 1),
- Learning the parameters of the opening recognition model from a **bigger dataset**,
- Learning the parameters of the opening recognition model from **bot vs bot replays**,
- Add $Opening^{t+1}$ and so the $P(Opening^{t+1}|Observations^{t})$ question explicitly.

Next? (1)

If we want a bot capable to compete at the highest level **against bots** (for 1.0 RC1):

- Most important: develop (with a release→benchmark loop) the full model (!),
- Add tactics (drops, run-by, contain...), and their counters,
- Vary tactics from learned outcome (enemy defenses type and position, UCT?),
- Abuse the game engine more (drop trick...),
- Deal with economy/tech/production and scouting/defense/attack concurrencies for resources/units (full model).

Next? (2)

If we want a bot capable to adapt to “good” **human play** (for v2.0):

- Dynamic adaptation of the strategy/build order, for instance through $P(BuildTree^{t+1}|Observations^{t}, BuildTree^{t})$ (see AIIDE 2011 for more informations),
- Detect fake builds,
- Detect fake tactical moves.

Bibliography

- Bayesian Robot Programming (2004) [Lebeltel O. *et al.*]
- Teaching Bayesian Behaviours to Video Game Characters (2004) [Le Hy R. *et al.*]
- A Data Mining Approach to Strategy Prediction (2009) [Weber B. & Mateas M.]
- Case-Based Planning and Execution for RTS Games (2007) [Ontañón S. *et al.*]
- Opponent Behaviour Recognition for Real-Time Strategy Games (2010) [Kabanza F. *et al.*]
- Building A Player Strategy Model by Analyzing Replays of Real-Time Strategy Games [Hsieh J-L. & Sun C-T.]
- Probability Theory: The Logic of Science (2003) [Jaynes E.T.]

Thanks

Thank you for your attention,

감사합니다

Questions ?