A Bayesian Tactician

Gabriel Synnaeve, Pierre Bessiere

A Bayesian Tactician

Gabriel Synnaeve (gabriel.synnaeve@gmail.com) and Pierre Bessière (pierre.bessiere@imag.fr)

Université de Grenoble (LIG), INRIA, CNRS, Collège de France (LPPA)

Abstract. We describe a generative Bayesian model of tactical attacks in strategy games, which can be used both to predict attacks and to take tactical decisions. This model is designed to easily integrate and merge information from other (probabilistic) estimations and heuristics. In particular, it handles uncertainty in enemy units' positions as well as their probable tech tree. We claim that learning, being it supervised or through reinforcement, adapts to skewed data sources. We evaluated our approach on StarCraft[1]: the parameters are learned on a new (freely available) dataset of game states, deterministically re-created from replays, and the whole model is evaluated for prediction in realistic conditions. It is also the tactical decision-making component of a competitive StarCraft AI.

1 Introduction

1.1 Game AI

We believe video game AI is central to new, fun, re-playable gameplays, being them multi-player or not. Cooperative (player versus AI) games are enjoying a new boom, recent RTS games delegate more micro-management to the AI as ever, and ever more realistic first-person shooters (FPS) immersion hardly cope with scripted (unsurprising) non-playing characters (NPC). In their study on human like characteristics in RTS games, Hagelbäck and Johansson [1] found out that "tactics was one of the most successful indicators of whether the player was human or not". No current non-cheating AI consistently beats good human players in RTS (aim cheating is harder to define for FPS games), nor are fun to play many games against. Finally, as the world is simulated but the players are not, multi-player game AI research is in between real-world robotics and more theoretical AI, and so can benefit both fields.

1.2 RTS Gameplay

Real-time strategy (RTS) gameplay consist in producing and managing group of units with attacks and movements specificities in order to defeat an enemy. Most often, it is required to gather resources and build up an economic and military power while expanding a technology tree. Parts of the map not in the sight range of the player's units are under *fog of war*, so the player only has partial information about the enemy buildings and army. The way by which we expand the tech tree, the specific units composing the army, and the general stance (aggressive or defensive) form what we call *strategy*. At the lower level, the actions performed by the player (human or not) to optimize the effectiveness of its units is called *micro-management*. In between lies *tactics*: where to attack, and how. A good human player takes much data in consideration when choosing: are there flaws in the defense? Which spot is more worthy to attack? How much am I vulnerable for attacking here? Is the terrain (height, chokes) to my advantage? etc.

In this paper, we focus on tactics, in between strategy (high-level) and micro-management (lower-level), as seen in Fig. 1. We propose a model which can either predict enemy attacks or give us a distribution on where and how to attack the opponent. Information from the higher-level strategy constrains what types of attacks are possible. As shown in Fig. 1, information from units' positions (or possibly an enemy units particle filter as in [2]) constrains where the armies can possibly be in the future. In the context of our StarCraft AI ("bot"), once we have a decision: we generate a goal (attack order) passed to units groups (see Fig.1). A Bayesian model for micro-management [3], in which units are attracted or repulsed by dynamic (goal, units, damages) and static (terrain) influence maps, actually moves the units in StarCraft. Other previous works on strategy prediction [4, 5] allows us to infer the enemy tech tree and strategies from incomplete information (due to the fog of war).

1.3 StarCraft Tactics

We worked on StarCraft: Brood War, which is a canonical RTS game. It had been around since 1998, sold 9.5 million licenses and reigned on competitive RTS for more than a decade. StarCraft (like most RTS) has a mechanism, *replays*, to record every

[1] StarCraft and its expansion StarCraft: Brood War are trademarks of Blizzard Entertainment™

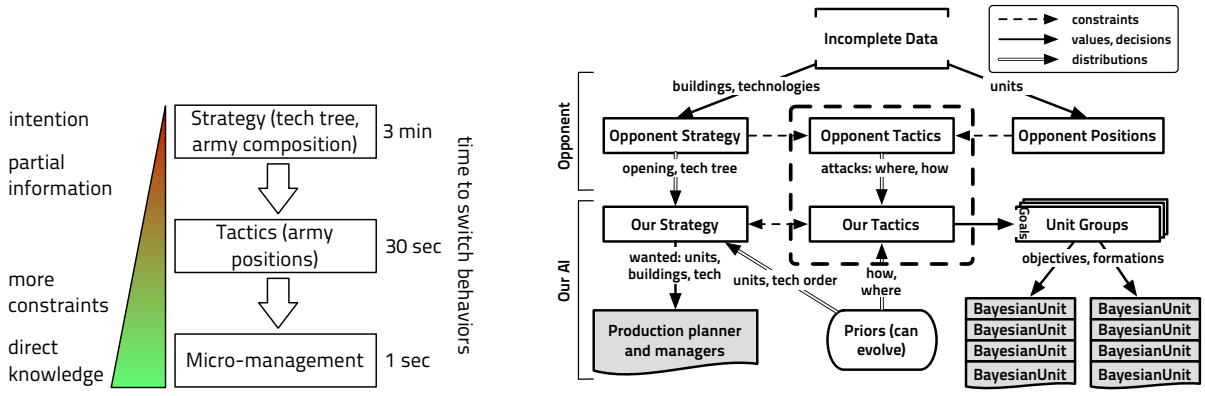

Fig. 1. Left: Gameplay levels of abstraction for RTS games, compared with their level of direct (and complete) information and orders of magnitudes of time to change their policies. Right:Information centric view of the StarCraft bot player, the part presented in this paper is inside dotted lines (tactics). Dotted arrows represent constraints on what is possible, plain simple arrows represent simple (real) values, either from data or decisions, and double arrows represent probability distributions on possible values. The grayed surfaces are the components actuators (passing orders to the game).

player's actions such that the state of the game can be deterministically re-simulated. Numerous international competitions and professional gaming (mainly in South Korea) produced a massive amount of data of highly skilled human players, performing about 300 actions per minute while following and adapting their strategies. In StarCraft, there are two types of resources, often located close together, minerals (at the base of everything) and gas (at the base of advanced units and technologies). There are 3 factions (Protoss, Terran and Zerg) which have workers to gather resources, and all other characteristics are different: from military units to "tech trees", gameplay styles.

Units have different abilities, which leads to different possible tactics. Each faction has invisible (temporarily or permanently) units, flying transport units, flying attack units and ground units. Some units can only attack ground or air units, some others have splash damage attacks, immobilizing or illusion abilities. Fast and mobile units are not cost-effective in head-to-head fights against slower bulky units. We used the gamers' vocabulary to qualify different types of tactics:

- *ground* attacks (raids or pushes) are the most normal kind of attacks, carried by basic units which cannot fly,
- *air* attacks (air raids), which use flying units' mobility to quickly deal damage to undefended spots.
- *invisible* attacks exploit the weaknesses (being them positional or technological) in detectors of the enemy to deal damage without retaliation,
- *drops* are attacks using ground units transported by air, combining flying units' mobility with cost-effectiveness of ground units, at the expense of vulnerability during transit.

This will be the only four types of tactics that we will use in this paper: *how* did the player attack or defend?

RTS games maps, StarCraft included, consist in a closed arena in which units can evolve. It is filled with terrain features like uncrossable terrain for ground units (water, space), cliffs, ramps, walls. Particularly, each RTS game which allows production also give some economical (gathering) mechanism and so there are some resources scattered on the map, where players need to go collect. It is way more efficient to build expansion (auxiliary bases) to collect resources directly on site. So when a player decides to attack, she has to decide *where* to attack, and this decision takes into account *how* it can attack different places, due to their geographical remoteness, topological access possibilities and defense strength. Choosing *where* to attack is a complex decision to make: of course it is always wanted to attack poorly defended economic expansions of the opponent, but the player has to consider if it places its own bases in jeopardy, or if it may trap her own army. With a perfect estimator of battles outcomes (which is a hard problem due to terrain, army composition combinatorics and units control complexity), and perfect information, this would result in a game tree problem which could be solved my $\alpha - \beta$. Unfortunately, StarCraft is a partial observation game with complex terrain and fight mechanics.

2 Background

2.1 Related Work

Aha et al. [6] used case-based reasoning (CBR) to perform dynamic tactical plan retrieval (matching) extracted from domain knowledge in Wargus. Ontañó et al. [7] based their real-time case-based planning (CBP) system on a plan dependency graph

which is learned from human demonstration in Wargus. A case based behavior generator spawn missing goals which are missing from the current state and plan according to the recognized state. In [8, 9], they used a knowledge-based approach to perform situation assessment to use the right plan, performing runtime adaptation by monitoring its performance. Sharma et al. [10] used richly parametrized CBR for strategic and tactical AI in Spring (Total Annihilation open source clone). [11] combined CBR and reinforcement learning to enable reuse of tactical plan components. Cadena and Garrido [12] used fuzzy CBR (fuzzy case matching) for strategic and tactical planning. Chung et al. [13] applied Monte-Carlo planning to a capture-the-flag mod of Open RTS. Inspired by successes of MCTS with upper confidence bounds on trees (UCT) policies [14] on the game of Go, Balla and Fern [15] applied UCT to tactical assault planning in Wargus.

In Starcraft, Weber et al. [16, 17] produced tactical goals through reactive planning and goal-driven autonomy, finding the more relevant goal(s) to follow in unforeseen situations. Kabanza et al. [18] performed plan and intent recognition to find tactical opportunities. On spatial and temporal reasoning, Forbus et al. [19] presented a tactical qualitative description of terrain for wargames through geometric and pathfinding analysis. Perkins [20] automatically extracted choke points and regions of StarCraft maps from a pruned Voronoi diagram, which we used for our regions representations. Wintermute et al. [21] used a cognitive approach mimicking human attention for tactics and units control. Ponsen et al. [22] developed an evolutionary state-based tactics generator for Wargus. Finally, Avery et al. [23] and Smith et al. [24] co-evolved influence map trees for spatial (tactical) reasoning in RTS games.

Our approach (and bot architecture, depicted in Fig. 1) can be seen as goal-driven autonomy [16] dealing with multi-level reasoning by passing distributions (without any assumption about how they were obtained) on the module input. Using distributions as messages between specialized modules makes dealing with uncertainty first class, this way a given model do not care if the uncertainty comes from incompleteness in the data, a complex and biased heuristic, or another probabilistic model. We then take a decision by sampling or taking the most probable value in the output distribution. Another particularity of our model is that it allows for prediction of the enemy tactics using the same model with different inputs. Finally, our approach is not exclusive to most of the techniques presented above, and it could be interesting to combine it with UCT [15] and more complex/precise tactics generated through planning.

2.2 Bayesian Programming

Probability is used as an alternative to classical logic and we transform incompleteness (in the experiences, observations or the model) into uncertainty [25]. We introduce Bayesian programs (BP), a formalism that can be used to describe entirely any kind of Bayesian model, subsuming Bayesian networks and Bayesian maps, equivalent to probabilistic factor graphs [26]. There are mainly two parts in a BP, the **description** of how to compute the joint distribution, and the **question(s)** that it will be asked.

The description consists in explaining the relevant *variables* $\{X^1, \ldots, X^n\}$ and explain their dependencies by *decomposing* the joint distribution $\mathrm{P}(X^1 \ldots X^n | \delta, \pi)$ with existing preliminary knowledge π and data δ. The *forms* of each term of the product specify how to compute their distributions: either parametric forms (laws or probability tables, with free parameters that can be learned from data δ) or recursive questions to other Bayesian programs.

Answering a question is computing the distribution $\mathrm{P}(Searched|Known)$, with $Searched$ and $Known$ two disjoint subsets of the variables.

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

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

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

Bayesian programming originated in robotics [27] and evolved to all sensory-motor systems [28]. For its use in cognitive modeling, see [29] and for its first use in video games (FPS, Unreal Tournament), see [30]; for MMORPG, see [31].

3 Methodology

3.1 Dataset

We downloaded more than 8000 replays to keep 7649 uncorrupted, 1v1 replays of very high level StarCraft games (pro-gamers leagues and international tournaments) from specialized websites[2][3][4], we then ran them using BWAPI[5] and dumped units positions, pathfinding and regions, resources, orders, vision events, for attacks (we trigger an attack tracking heuristic when one unit dies and there are at least two military units around): types, positions, outcomes. Basically, every BWAPI event was recorded, the dataset and its source code are freely available[6].

3.2 Spatial Reasoning

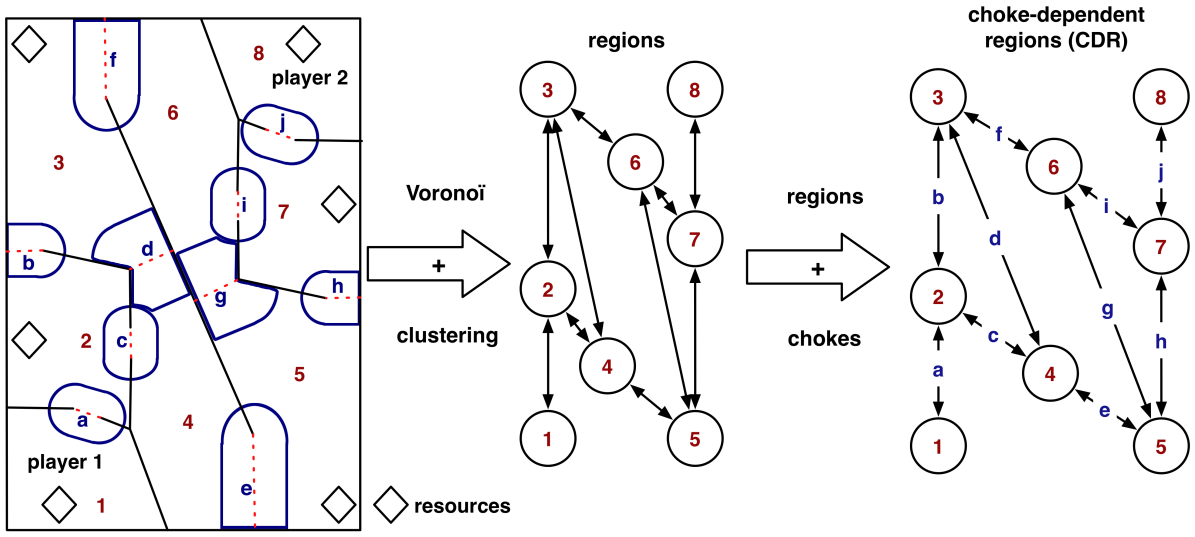

Fig. 2. A very simple map on the left, which is transformed into regions (between chokes in dotted red lines) by Voronoi tesselation and clustering. These plain regions (numbers in red) are then augmented with choke-dependent regions (letters in blue)

We used two kinds of regions: BroodWar Terrain Analyser (BWTA) regions and choke-dependent (choke-centered) regions. BWTA regions are obtained from a pruned Voronoi diagram on walkable terrain [20] and give regions for which chokes are the boundaries. As battles often happens at chokes, choke-dependent regions are created by doing an additional (distance limited) Voronoi tesselation spawned at chokes, its regions set is $(regions \setminus chokes) \cup chokes$. Figure 2 illustrate regions and choke-dependent regions (CDR). Results for choke-dependent regions are not fully detailed.

3.3 Value Heuristics

The idea is to have (most probably biased) lower-level heuristics from units observations which produce information exploitable at the tactical level, and take some advantage of strategic inference too. The advantages are that 1) learning will de-skew the model output from biased heuristic inputs 2) the model is agnostic to where input variables' values come from 3) the updating process is the same for supervised learning and for reinforcement learning.

We note $s^{a|d}_{unit\ type}(r)$ for the balanced score of units from attacker or defender ($^{a|d}$) of a given type in region r. The balanced score of units is just the sum of units multiplied by each unit score ($= minerals\_value + \frac{4}{3} gas\_value + 50 supply\_value$).

[2] http://www.teamliquid.net
[3] http://www.gosugamers.net
[4] http://www.iccup.com
[5] http://code.google.com/p/bwapi/
[6] http://snippyhollow.github.com/bwrepdump/

The heuristics we used in our benchmarks (which we could change) are:

$$economical\_score^d(r) = \frac{s^d_{workers}(r)}{\sum_{i \in regions} s^d_{workers}(i)}$$

$$tactical\_score^d(r) = \sum_{i \in regions} s^d_{army}(i) \times dist(i,r)^{-1.5}$$

We used "-1.5" such that the tactical value of a region in between two halves of an army, each at distance 2, would be higher than the tactical value of a region at distance 4 of the full (same) army. For flying units, *dist* is the Euclidean distance, while for ground units it takes pathfinding into account.

$$ground\_defense^d(r) = \frac{s^d_{can\_attack\_ground}(r)}{s^a_{ground\_units}(r)}$$

$$air\_defense^d(r) = \frac{s^d_{can\_attack\_air}(r)}{s^a_{air\_units}(r)}$$

$$invis\_defense^d(r) = number^d_{detectors}$$

3.4 Tactical Model

We preferred to discretize continuous values to enable quick complete computations. Another strategy would keep more values and use Monte Carlo sampling for computation. We think that discretization is not a concern because 1) heuristics are simple and biased already 2) we often reason about imperfect information and this uncertainty tops discretization fittings.

Variables With n regions, we have:

- $A_{1:n} \in \{true, false\}$, A_i: attack in region i or not?
- $E_{1:n} \in \{no, low, high\}$, E_i is the discretized economical value of the region i for the defender. We choose 3 values: *no* workers in the regions, *low*: a small amount of workers (less than half the total) and *high*: more than half the total of workers in this region i.
- $T_{1:n} \in discrete\ levels$, T_i is the tactical value of the region i for the defender, see above for an explanation of the heuristic. Basically, T is proportional to the proximity to the defender's army. In benchmarks, discretization steps are $0, 0.05, 0.1, 0.2, 0.4, 0.8$ (log_2 scale).
- $TA_{1:n} \in discrete\ levels$, TA_i is the tactical value of the region i for the attacker (see above).
- $B_{1:n} \in \{true, false\}$, B_i tells if the region belongs (or not) to the defender. $\mathrm{P}(B_i = true) = 1$ if the defender has a base in region i and $\mathrm{P}(B_i = false) = 1$ if the attacker has one. Influence zones of the defender can be measured (with uncertainty) by $\mathrm{P}(B_i = true) \geq 0.5$ and vice versa.
- $H_{1:n} \in \{ground, air, invisible, drop\}$, H_i: in predictive mode: how we will be attacked, in decision-making: how to attack, in region i.
- $GD_{1:n} \in \{no, low, med, high\}$: ground defense (relative to the attacker power) in region i, result from a heuristic. *no* defense if the defender's army is $\geq 1/10th$ of the attacker's, *low* defense above that and under half the attacker's army, *medium* defense above that and under comparable sizes, *high* if the defender's army is bigger than the attacker.
- $AD_{1:n} \in \{no, low, med, high\}$: same for air defense.
- $ID_{1:n} \in \{no\ detector, one\ detector, several\}$: invisible defense, equating to numbers of detectors.
- $TT \in [\emptyset, building_1, building_2, building_1 \wedge building_2, techtrees, \dots]$: all the possible technological trees for the given race. For instance $\{pylon, gate\}$ and $\{pylon, gate, core\}$ are two different Tech Trees.
- $HP \in \{ground, ground \wedge air, ground \wedge invis, ground \wedge air \wedge invis, ground \wedge drop, ground \wedge air \wedge drop, ground \wedge invis \wedge drop, ground \wedge air \wedge invis \wedge drop\}$: **h**ow **p**ossible types of attacks, directly mapped from TT information. In prediction, with this variable, we make use of what we can infer on the opponent's strategy [5, 4], in decision-making, we know our own possibilities (we know our tech tree as well as the units we own).

Finally, for some variables, we take uncertainty into account with "soft evidences": for instance for a region in which no player has a base, we have a soft evidence that it belongs more probably to the player established closer. In this case, for a given region, we introduce the soft evidence variable(s) B' and the coherence variable λ_B and impose $\mathrm{P}(\lambda_B = 1|B, B')\ iff\ B = B'$, while $\mathrm{P}(\lambda_B|B, B')\mathrm{P}(B')$ is a new factor in the joint distribution. This allows to sum over $\mathrm{P}(B')$ distribution (soft evidence).

Decomposition The joint distribution of our model contains soft evidence variables for all input family variables (E, T, TA, B, GD, AD, ID, HP) to be as general as possible, *i.e.* to be able to cope with all possible uncertainty (from incomplete information) that may come up in a game. To avoid being too verbose, we explain the decomposition only with the soft evidence for the family of variables B, the principle holds for all other soft evidences. For the n considered regions, we have:

$$\begin{aligned}
&\mathrm{P}(A_{1:n}, E_{1:n}, T_{1:n}, TA_{1:n}, B_{1:n}, B'_{1:n}, \lambda_{B,1:n},\\
&H_{1:n}, GD_{1:n}, AD_{1:n}, ID_{1:n}, HP, TT)\\
=&\prod_{i=1}^{n}[\mathrm{P}(A_i)\mathrm{P}(E_i, T_i, TA_i, B_i|A_i). \qquad (1)\\
&\mathrm{P}(\lambda_{B,i}|B_{1:n}, B'_{1:n})\mathrm{P}(B'_{1:n})\\
&\mathrm{P}(AD_i, GD_i, ID_i|H_i)\mathrm{P}(H_i|HP)]\mathrm{P}(HP|TT)\mathrm{P}(TT)
\end{aligned}$$

Forms and Learning We will explain the forms for a given/fixed i region number:

- $\mathrm{P}(A)$ is the prior on the fact that the player attacks in this region, in our evaluation we set it to $n_{battles}/(n_{battles} + n_{not\ battles})$. In a given match it should be initialized to uniform and progressively learn the preferred attack regions of the opponent for predictions, learn the regions in which our attacks fail or succeed for decision-making.
- $\mathrm{P}(E, T, TA, B|A)$ is a covariance table of the economical, tactical (both for the defender and the attacker), belonging scores where an attacks happen. We just use Laplace succession law ("add one" smoothing) [25] and count the co-occurrences, thus almost performing maximum likelihood learning of the table.
- $\mathrm{P}(\lambda_B|B, B') = 1.0\ iff\ B = B'$ is just a coherence constraint.
- $\mathrm{P}(AD, GD, ID|H)$ is a covariance table of the air, ground, invisible defense values depending on how the attack happens. As for $\mathrm{P}(E, T, TA, B|A)$, we use a Laplace's law of succession to learn it.
- $\mathrm{P}(H|HP)$ is the distribution on how the attack happens depending on what is possible. Trivially $\mathrm{P}(H = ground|HP = ground) = 1.0$, for more complex possibilities we have different maximum likelihood multinomial distributions on H values depending on HP.
- $\mathrm{P}(HP|TT)$ is the direct mapping of what the tech tree allows as possible attack types: $\mathrm{P}(HP = hp|TT) = 1$ is a function of TT (all $\mathrm{P}(HP \neq hp|TT) = 0$).
- $\mathrm{P}(TT)$: if we are sure of the tech tree (prediction without fog of war, or in decision-making mode), $\mathrm{P}(TT = k) = 1$ and $\mathrm{P}(TT \neq k) = 0$; otherwise, it allows us to take uncertainty about the opponent's tech tree and balance $\mathrm{P}(HP|TT)$. We obtain a distribution on what is possible ($\mathrm{P}(HP)$) for the opponent's attack types.

There are two approaches to fill up these probability tables, either by observing games (supervised learning), as we did in the evaluation section, or by acting (reinforcement learning). In match situation against a given opponent, for inputs that we can unequivocally attribute to their intention (style and general strategy), we also refine these probability tables (with Laplace's rule of succession). To keep things simple, we just refine $\sum_{E,T,TA} \mathrm{P}(E, T, TA, B|A)$ corresponding to their aggressiveness (aggro) or our successes and failures, and equivalently for $\mathrm{P}(H|HP)$. Indeed, if we sum over E, T and TA, we consider the inclination of our opponent to venture into enemy territory or the interest that we have to do so by counting our successes with aggressive or defensive parameters. In $\mathrm{P}(H|HP)$, we are learning the opponent's inclination for particular types of tactics according to what is available to their, or for us the effectiveness of our attack types choices.

The model is highly modular, and some parts are more important than others. We can separate three main parts: $\mathrm{P}(E, T, TA, B|A)$, $\mathrm{P}(AD, GD, ID|H)$ and $\mathrm{P}(H|HP)$. In prediction, $\mathrm{P}(E, T, TA, B|A)$ uses the inferred (uncertain) economic (E), tactical (T) and belonging (B) scores of the opponent while knowing our own tactical position fully (TA). In decision-making, we know E, T, B (for us) and estimate TA. In our prediction benchmarks, $\mathrm{P}(AD, GD, ID|H)$ has the lesser impact on the results of the three main parts, either because the uncertainty from the attacker on AD, GD, ID is too high or because our heuristics are too simple, though it still contributes positively to the score. In decision-making, it allows for reinforcement learning to have pivoting tuple values for AD, GD, ID at which to switch attack types. In prediction, $\mathrm{P}(H|HP)$ is used to take $\mathrm{P}(TT)$ (coming from strategy prediction [4]) into account and constraints H to what is possible. For the use of $\mathrm{P}(H|HP)\mathrm{P}(HP|TT)\mathrm{P}(TT)$ in decision-making, see the Results sections.

Questions For a given region i, we can ask the probability to attack here,

$$\mathrm{P}(A_i = a_i|e_i, t_i, ta_i, \lambda_{B,i} = 1)$$

$$= \frac{\sum_{B_i,B'_i} \mathrm{P}(e_i,t_i,ta_i,B_i|a_i)\mathrm{P}(a_i)\mathrm{P}(B'_i).P(\lambda_{B,i}|B_i,B'_i)}{\sum_{A_i,B_i,B'_i} \mathrm{P}(e_i,t_i,ta_i,B_i|A_i)\mathrm{P}(A_i)\mathrm{P}(B'_i)\mathrm{P}(\lambda_{B,i}|B_i,B'_i)}$$
$$\propto \sum_{B_i,B'_i} \mathrm{P}(e_i,t_i,ta_i,B_i|a_i)\mathrm{P}(a_i)\mathrm{P}(B'_i)\mathrm{P}(\lambda_{B,i}|B_i,B'_i)$$

and the mean by which we should attack,

$$\mathrm{P}(H_i = h_i|ad_i,gd_i,id_i)$$
$$\propto \sum_{TT,P} [\mathrm{P}(ad_i,gd_i,id_i|h_i)\mathrm{P}(h_i|HP)\mathrm{P}(HP|TT)\mathrm{P}(TT)]$$

For clarity, we omitted some variables couples on which we have to sum (to take uncertainty into account) as for B (and B') above. We always sum over estimated, inferred variables, while we know the one we observe fully. In prediction mode, we sum over TA, B, TT, P; in decision-making, we sum over E, T, B, AD, GD, ID. The complete question that we ask our model is $\mathrm{P}(A, H|FullyObserved)$. The maximum of $\mathrm{P}(A, H)$ may not be the same as the maximum of $\mathrm{P}(A)$ or $\mathrm{P}(H)$, for instance think of a very important economic zone that is very well defended, it may be the maximum of $\mathrm{P}(A)$, but not once we take $\mathrm{P}(H)$ into account. Inversely, some regions are not defended against anything at all but present little or no interest. Our joint distribution (1) can be rewritten: $\mathrm{P}(Searched, FullyObserved, Estimated)$, so we ask:

$$\mathrm{P}(A_{1:n}, H_{1:n}|FullyObserved) \qquad (2)$$
$$\propto \sum_{Estimated} \mathrm{P}(A_{1:n}, H_{1:n}, Estimated, FullyObserved)$$

4 Results

4.1 Learning

To measure fairly the prediction performance of such a model, we applied "leave-100-out" cross-validation from our dataset: as we had many games (see Table. 1), we set aside 100 games of each match-up for testing (with more than 1 battle per match: rather $\approx [\![11 \dots 35]\!]$ battles/match) and train our model on the rest. We write match-ups XvY with X and Y the first letters of the factions involved (Protoss, Terran, Zerg). Note that mirror match-ups (PvP, TvT, ZvZ) have fewer games but twice as many attacks from a given faction. Learning was performed as explained in III.B.3: for each battle in r we had one observation for: $\mathrm{P}(e_r, t_r, ta_r, b_r|A = true)$, and $\#regions - 1$ observations for the i regions which were not attacked: $\mathrm{P}(e_{i\neq r}, t_{i\neq r}, ta_{i\neq r}, b_{i\neq r}|A = false)$. For each battle of type t we had one observation for $P(ad, gd, id|H = t)$ and $P(H = t|p)$. By learning with a Laplace's law of succession [25], we allow for unseen event to have a non-zero probability.

An exhaustive presentation of the learned tables is out of the scope of this paper, but we displayed interesting cases in which the posteriors of the learned model concur with human expertise in Figures 3 and 4. In Fig. 3, we see that air raids/attacks are quite risk averse and it is two times more likely to attack a region with less than 1/10th of the flying force in anti-aircraft warfare than to attack a region with up to one half of our force. We can also notice than drops are to be preferred either when it is safe to land (no anti-aircraft defense) or when there is a large defense (harassment tactics). In Fig. 3 we can see that, in general, there are as many ground attacks at the sum of other types. The two top graphs show cases in which the tech of the attacker was very specialized, and, in such cases, the specificity seems to be used. In particular, the top right graphic may be corresponding to a "fast Dark Templars rush". Finally, Fig. 4 shows the transition between two types of encounters: tactics aimed at engaging the enemy army (a higher T value entails a higher $\mathrm{P}(A)$) and tactics aimed at damaging the enemy economy (at high E, we look for opportunities to attack with a small army where T is lower).

4.2 Prediction Performance

We learned and tested one model for each race and each match-up. As we want to predict *where* ($\mathrm{P}(A_{1:n})$) and *how* ($\mathrm{P}(H_{battle})$) the next attack will happen to us, we used inferred enemy TT (to produce P) and TA, our scores being fully known: E, T, B, ID. We consider GD, AD to be fully known even though they depend on the attacker force, we should have some uncertainty on them, but we tested that they accounted (being known instead of fully unknown) for 1 to 2% of $\mathrm{P}(H)$ accuracy (in prediction) once HP was known. We should point that pro-gamers scout very well and so it allows for a highly accurate TT estimation with [4]. Training requires to recreate battle states (all units' positions) and count parameters for 5000 to 30000 battles (depending on the match-up). Once that is done, inference is very quick: a look-up in a probability table for known values and $\#F$ look-ups for free variables F on which we sum. We chose to try and predict the next battle 30 seconds before it happens, 30 seconds

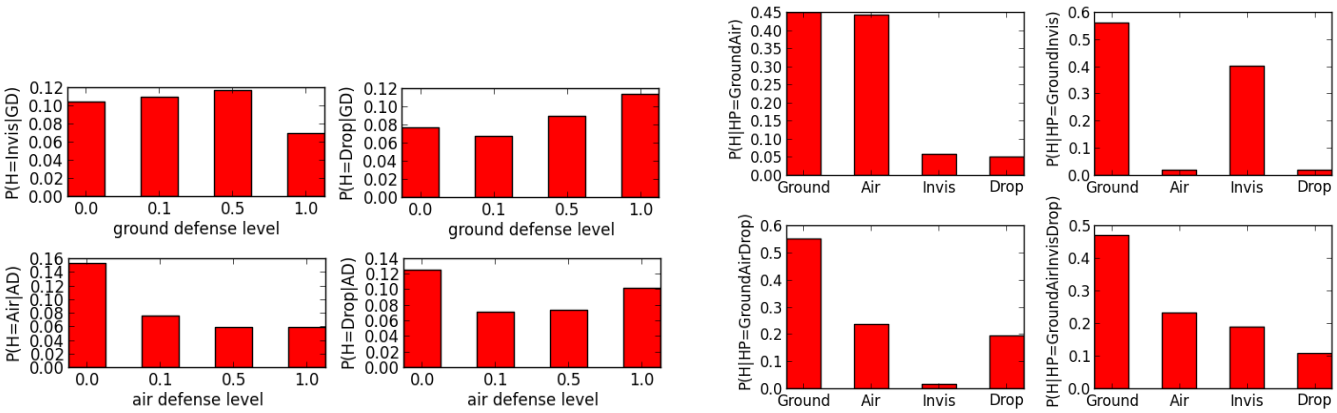

Fig. 3. Left: (top) $\mathrm{P}(H = invis)$ and $\mathrm{P}(H = drop)$ for varying values of GD (summed on other variables); (bottom) $\mathrm{P}(H = air)$ and $\mathrm{P}(H = drop)$ for varying values of AD (summed on other variables), for Terran in TvP. We can see that it is far more likely that invisible ("sneaky") attacks happen where there is low ground presence (top left plot). For drops, we understand that the high value for $\mathrm{P}(H = drop|GD = 1.0)$ is caused by the fact that drop armies are small and this value corresponds to drops which are being expected by the defender. Drops at lower values of GD correspond to unexpected (surprise) drops. As ground units are more cost efficient than flying units in a static battle, we see that both $\mathrm{P}(H = invis|AD = 0.0)$ and $\mathrm{P}(H = drop|AD = 0.0)$ are much more probable than situations with air defenses. Right: $\mathrm{P}(H|HP)$ for varying values of H and for different values of P (derived from inferred TT), for Protoss in PvT. Conditioning on what is possible given the *tech tree* gives a lot of information about what attack types are possible or not. More interestingly, it clusters the game phases in different tech levels and allows for learning the relative distributions of attack types with regard to each phase. For instance, the last (bottom right) plot shows the distribution on attack types at the end of a technologically complete game.

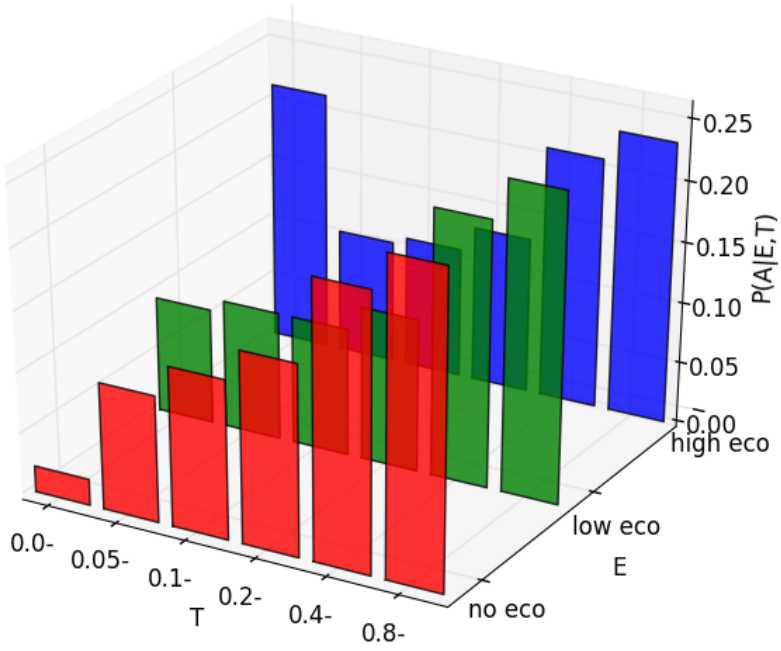

Fig. 4. $\mathrm{P}(A)$ for varying values of E and T, summed on the other variables, for Terran in TvT. Higher economical values is strongly correlated with surprise attacks with small tactical squads and no defenses, which almost never happens in open fields ("no eco") as this would lead to very unbalanced battles (in terms of army sizes): it would not benefit the smaller party, which can flee and avoid confrontation, as opposed to when defending their base.

being an approximation of the time needed to go from the middle of a map (where the entropy on "next battle position" is maximum) to any region by ground, so that the prediction is useful for the defender (they can position their army).

The model code[7] (for learning and testing) as well as the datasets (see above) are freely available. Raw results of predictions of positions and types of attacks 30 seconds before they happen are presented in Table. 1: for instance the bold number (38.0) corresponds to the percentage of good positions (regions) predictions (30 sec before event) which were ranked 1st in the probabilities on $A_{1:n}$ for Protoss attacks against Terran (PvT). The measures on *where* corresponds to the percentage of good prediction and the mean probability for given ranks in $\mathrm{P}(A_{1:n})$ (to give a sense of the shape of the distribution). As the most probable The measures on *how* corresponds to the percentage of good predictions for the most probable $\mathrm{P}(H_{battle})$ and the number of such battles seen in the test set for given attack types. We particularly predict well ground attacks (trivial in the early game, less in the end game) and, interestingly, Terran and Zerg drop attacks. The *where & how* row corresponds to the percentage of good predictions for the maximal probability in the joint $\mathrm{P}(A_{1:n}, H_{1:n})$: considering only the most probable attack (more information is in the rest of the distribution, as shown for *where*!) according to our model, we can predict *where* **and** *how* an attack will occur in the next 30 seconds $\approx$ 1/4th of the time. Finally, note that scores are also useful 60 seconds before the attack (obviously, TT, and thus P, are not so different, nor are B and E): PvT *where* top 4 ranks are 35.6, 8.5, 7.7, 7.0% good versus 38.0, 16.3, 8.9, 6.7% 30 seconds before; *how* total precision 60 seconds before is 70.0% vs. 72.4%, *where & how* maximum probability precision is 19.9% vs. 23%. This gives even more time for the player to adapt their tactics.

Table 1. Results summary for multiple metrics at 30 seconds before attack. The number in bold (38.0) is read as "38% of the time, the region i with probability of rank 1 in $\mathrm{P}(A_i)$ is the one in which the attack happened 30 seconds later".

%: good predictions		Protoss						Terran						Zerg					
Pr: mean probability		P		T		Z		P		T		Z		P		T		Z	
total # games		445		2408		2027		2408		461		2107		2027		2107		199	
measure	rank	%	Pr	%	Pr	%	Pr	%	Pr	%	Pr	%	Pr	%	Pr	%	Pr	%	Pr
where	1	40.9	.334	**38.0**	.329	34.5	.304	35.3	.299	34.4	.295	39.0	0.358	32.8	.31	39.8	.331	37.2	.324
	2	14.6	.157	16.3	.149	13.0	.152	14.3	.148	14.7	.147	17.8	.174	15.4	.166	16.6	.148	16.9	.157
	3	7.8	.089	8.9	.085	6.9	.092	9.8	.09	8.4	.087	10.0	.096	11.3	.099	7.6	.084	10.7	.100
	4	7.6	.062	6.7	.059	7.9	.064	8.6	.071	6.9	.063	7.0	.062	8.9	.07	7.7	.064	8.6	.07
measure	type	%	N	%	N	%	N	%	N	%	N	%	N	%	N	%	N	%	N
how	G	97.5	1016	98.1	1458	98.4	568	100	691	99.9	3218	76.7	695	86.6	612	99.8	567	67.2	607
	A	44.4	81	34.5	415	46.8	190	40	5	13.3	444	47.1	402	14.2	155	15.8	19	74.2	586
	I	22.7	225	49.6	337	12.9	132	NA	NA	NA	NA	36.8	326	32.6	227	NA	NA	NA	NA
	D	55.9	340	42.2	464	45.2	93	93.5	107	86	1183	62.8	739	67.7	535	81.4	86	63.6	588
total		76.3	1662	72.4	2674	71.9	983	98.4	806	88.5	4850	60.4	2162	64.6	1529	94.7	674	67.6	1802
where & how (%)		32.8		23		23.8		27.1		23.6		30.2		23.3		30.9		26.4	

When we are mistaken, the mean ground distance (pathfinding wise) of the most probable predicted region to the good one (where the attack happens) is 1223 pixels (38 build tiles, or 2 screens in StarCraft's resolution), while the mean max distance on the map is 5506 (172 build tiles). Also, the mean number of regions by map is 19, so a random *where* (attack destination) picking policy would have a correctness of 1/19 (5.23%). For choke-centered regions, the numbers of good *where* predictions are lower (between 24% and 32% correct for the most probable) but the mean number of regions by map is 42. For *where & how*, a random policy would have a precision of 1/(19*4), and even a random policy taking the high frequency of ground attacks into account would at most be $\approx$ 1/(19*2) correct.

For the location only (*where* question), we also counted the mean number of different regions which were attacked in a given game (between 3.97 and 4.86 for regions, depending on the match-up, and between 5.13 and 6.23 for choke-dependent regions). The ratio over these means would give the best prediction rate we could expect from a *baseline heuristic* based solely on the location data. These are attacks that actually happened, so the number of regions a player have to be worried about is at least this one (or more, for regions which were not attacked during a game but were potential targets). This *baseline heuristic* would yield (depending on the match-up) prediction rates between 20.5 and 25.2% for regions, versus our 32.8 to 40.9%, and between 16.1% and 19.5% for choke-dependent regions, versus our 24% to 32%.

Note that our current model considers a uniform prior on regions (no bias towards past battlefields) and that we do not incorporate any derivative of the armies' movements. There is no player modeling at all: learning and fitting the mean player's tactics is not optimal, so we should specialize the probability tables for each player. Also, we use all types of battles in our training and testing. Short experiments showed that if we used only attacks on bases, the probability of good *where* predictions for the maximum of $\mathrm{P}(A_{1:n})$ goes above 50% (which is not a surprise, there are far less bases than regions in which attacks happen). To conclude on tactics positions prediction: if we sum the 2 most probable regions for the attack, we are right at least half the time; if we sum the 4 most probable (for our robotic player, it means it prepares against attacks in 4 regions as opposed to 19), we are right $\approx$ **70%** of the time.

[7] `https://github.com/SnippyHolloW/AnalyzeBWData`

Mistakes on the type of the attack are high for invisible attacks: while these tactics can definitely win a game, the counter is strategic (it is to have detectors technology deployed) more than positional. Also, if the maximum of $\mathrm{P}(H_{battle})$ is wrong, it doesn't mean than $\mathrm{P}(H_{battle} = good) = 0.0$ at all! The result needing improvements the most is for air tactics, because countering them really is positional, see our discussion in the conclusion.

4.3 In Game Decision-Making

In a StarCraft game, our bot has to make decisions about where and how to attack or defend, it does so by reasoning about opponent's tactics, bases, its priors, and under strategic constraints (Fig. 1). Once a decision is taken, the output of the tactical model is an offensive or defensive goal. There are different military goal types (base defense, ground attacks, air attacks, drops...), and each type of goal has pre-requisites (for instance: a drop goal needs to have the control of a dropship and military units to become active). The spawned goal then autonomously sets objectives for Bayesian units [3], sometimes procedurally creating intermediate objectives or canceling itself in the worst cases.

The destinations of goals are from $\mathrm{P}(A)$, while the type of the goal comes from $\mathrm{P}(H)$. In input, we fully know tactical scores of the regions according to our military units placement TA (we are the attacker), what is possible for us to do P (according to units available) and we estimate E, T, B, ID, GD, AD from past (partial) observations. Estimating T is the most tricky of all because it may be changing fast, for that we use a units filter which just decays probability mass of seen units. An improvement would be to use a particle filter [2], with a learned motion model. From the joint (2) $\mathrm{P}(A_{1:n}, H_{1:n}|ta, p, tt)$ may arise a couple i, H_i more probable than the most probables $\mathrm{P}(A_i)$ and $\mathrm{P}(H_j)$ taken separately (the case of an heavily defended main base and a small unprotected expand for instance). Fig. 5 displays the mean $\mathrm{P}(A, H)$ for Terran (in TvZ) attacks decision-making for the most 32 probable type/region tactical couples. It is in this kind of landscape (though more steep because Fig. 5 is a mean) that we sample (or pick the most probable couple) to take a decision. Also, we may spawn defensive goals countering the attacks that we predict from the opponent.

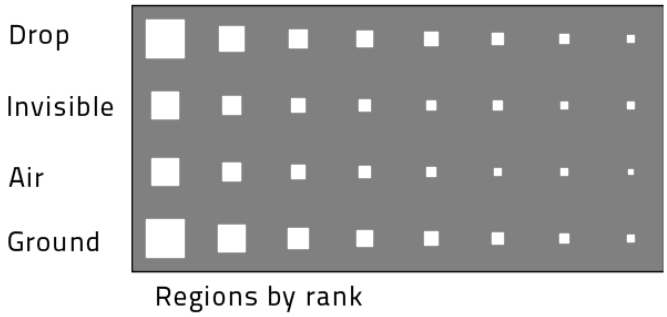

Fig. 5. Mean $\mathrm{P}(A, H)$ for all H values and the top 8 $\mathrm{P}(A_i, H_i)$ values, for Terran in TvZ. The larger the white square area, the higher $\mathrm{P}(A_i, H_i)$. A simple way of taking a tactical decision according to this model, and the learned parameters, is by sampling in this distribution.

Finally, we can steer our technological growth towards the opponent's weaknesses. A question that we can ask our model (at time t) is $\mathrm{P}(TT)$, or, in two parts: we first find i, h_i which maximize $\mathrm{P}(A, H)$ at time $t + 1$, and then ask a more directive:

$$\mathrm{P}(TT|h_i) \propto \sum_P \mathrm{P}(h_i|HP)\mathrm{P}(P|TT)\mathrm{P}(TT)$$

so that it gives us a distribution on the tech trees (TT) needed to be able to perform the wanted attack type. To take a decision on our technology direction, we can consider the distances between our current tt^t and all the probable values of TT^{t+1}.

5 Conclusions

5.1 Possible Improvements

There are three main research directions for possible improvements: improving the underlying heuristics, improving the dynamic of the model and improving the model itself. The heuristics presented here are quite simple but they may be changed, and even removed or added, for another RTS or FPS, or for more performance. In particular, our "defense against invisible" heuristic

could take detector positioning/coverage into account. Our heuristic on tactical values can also be reworked to take terrain tactical values into account (chokes and elevation in StarCraft). For the estimated position of enemy units, we could use a particle filter [2] with a motion model (at least one for ground units and one for flying units). There is room to improve the dynamics of the model: considering the prior probabilities to attack in regions given past attacks and/or considering evolutions of the T,TA,B,E values (derivatives) in time. The discretizations that we used may show their limits, though if we want to use continuous values, we need to setup a more complicated learning and inference process (MCMC sampling). Finally, one of the strongest assumptions (which is a drawback particularly for prediction) of our model is that the attacking player is always considered to attack in this most probable regions. While this would be true if the model was complete (with finer army positions inputs and a model of what the player thinks), we believe such an assumption of completeness is far fetched. Instead we should express that incompleteness in the model itself and have a "player decision" variable $D \sim Multinomial(\mathrm{P}(A_{1:n}, H_{1:n}), player)$.

5.2 Final Words

We have presented a Bayesian tactical model for RTS AI, which allows both for opposing tactics prediction and autonomous tactical decision-making. Being a probabilistic model, it deals with uncertainty easily, and its design allows easy integration into multi-granularity (multi-scale) AI systems as needed in RTS AI. Without any temporal dynamics, its exact prediction rate of the joint position and tactical type is in [23-32.8]% (depending on the match-up), and considering the 4 most probable regions it goes up to $\approx 70\%$. More importantly, it allows for tactical decision-making under (technological) constraints and (state) uncertainty. It can be used in production thanks to its low CPU and memory footprint. The dataset, its documentation[8], as well as our model implementation[9] (and other data-exploration tools) are free software and can be found online. We plan to use this model in our StarCraft AI competition entry bot as it gives our bot tactical autonomy and a way to adapt to our opponent.

References

1. Hagelbäck, J., Johansson, S.J.: A Study on Human like Characteristics in Real Time Strategy Games. In: Proceedings of CIG, IEEE (2010)
2. Weber, B.G., Mateas, M., Jhala, A.: A Particle Model for State Estimation in Real-Time Strategy Games. In: Proceedings of AIIDE, Stanford, Palo Alto, California, AAAI Press (2011) 103–108
3. Synnaeve, G., Bessière, P.: A Bayesian Model for RTS Units Control applied to StarCraft. In: Proceedings of CIG, Seoul, South Korea, IEEE (September 2011)
4. Synnaeve, G., Bessière, P.: A Bayesian Model for Plan Recognition in RTS Games applied to StarCraft. In: Proceedings of AIIDE, Palo Alto, CA, USA, AAAI Press (October 2011) 79–84
5. Synnaeve, G., Bessière, P.: A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft. In: Proceedings of CIG, Seoul, South Korea, IEEE (September 2011)
6. Aha, D.W., Molineaux, M., Ponsen, M.J.V.: Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. In: ICCBR. (2005) 5–20
7. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Proceedings of the 7th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development. International Joint Conference on Neural Networks (ICCBR-07), Springer-Verlag (2007) 164–178
8. Mishra, K., Ontañón, S., Ram, A.: Situation Assessment for Plan Retrieval in Real-Time Strategy Games. In: ECCBR. (2008) 355–369
9. Meta, M., Ontañón, S., Ram, A.: Meta-Level Behavior Adaptation in Real-Time Strategy Games. In: ICCBR-10 Workshop on Case-Based Reasoning for Computer Games, Alessandria, Italy. (2010)
10. Bakkes, S.C.J., Spronck, P.H.M., Jaap van den Herik, H.: Opponent modelling for case-based adaptive game AI. Entertainment Computing **1**(1) (January 2009) 27–37
11. Sharma, M., Holmes, M., , Santamaria, J., Irani, A., Isbell, C.L., Ram, A.: Transfer Learning in Real-Time Strategy Games Usinging Hybrid CBR/RL. In: International Joint Conference of Artificial Intelligence, IJCAI. (2007)
12. Cadena, P., Garrido, L.: Fuzzy Case-Based Reasoning for Managing Strategic and Tactical Reasoning in StarCraft. In: Proceedings of MICAI (1), Springer (2011) 113–124
13. Chung, M., Buro, M., Schaeffer, J.: Monte Carlo Planning in RTS Games. In: Proceedings of CIG, IEEE (2005)
14. Gelly, S., Wang, Y.: Exploration exploitation in Go: UCT for Monte-Carlo Go. In: NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop, Canada (December 2006)
15. Balla, R.K., Fern, A.: UCT for Tactical Assault Planning in Real-Time Strategy Games. In: IJCAI. (2009)
16. Weber, B.G., Mateas, M., Jhala, A.: Applying Goal-Driven Autonomy to StarCraft. In: Artificial Intelligence and Interactive Digital Entertainment (AIIDE). (2010)
17. Weber, B.G., Mawhorter, P., Mateas, M., Jhala, A.: Reactive Planning Idioms for Multi-Scale Game AI. In: Proceedings of CIG, IEEE (2010)

[8] `http://snippyhollow.github.com/bwrepdump/`
[9] `https://github.com/SnippyHolloW/AnalyzeBWData`

18. Kabanza, F., Bellefeuille, P., Bisson, F., Benaskeur, A.R., Irandoust, H.: Opponent Behaviour Recognition for Real-Time Strategy Games. In: AAAI Workshops. (2010)
19. Forbus, K.D., Mahoney, J.V., Dill, K.: How qualitative spatial reasoning can improve strategy game ais. IEEE Intelligent Systems **17** (July 2002) 25–30
20. Perkins, L.: Terrain Analysis in Real-Time Strategy Games: An Integrated Approach to Choke Point Detection and Region Decomposition. In: Proceedings of AIIDE, AAAI Press (2010)
21. Wintermute, S., Joseph Xu, J.Z., Laird, J.E.: SORTS: A Human-Level Approach to Real-Time Strategy AI. In: Proceedings of AIIDE, AAAI Press (2007) 55–60
22. Ponsen, M.J.V., Muñoz-Avila, H., Spronck, P., Aha, D.W.: Automatically Generating Game Tactics through Evolutionary Learning. AI Magazine **27**(3) (2006) 75–84
23. Avery, P., Louis, S., Avery, B.: Evolving Coordinated Spatial Tactics for Autonomous Entities using Influence Maps. In: Proceedings of the 5th international conference on Computational Intelligence and Games. Proceedings of CIG, Piscataway, NJ, USA, IEEE (2009) 341–348
24. Smith, G., Avery, P., Houmanfar, R., Louis, S.: Using Co-evolved RTS Opponents to Teach Spatial Tactics. In: Proceedings of CIG, IEEE (2010)
25. Jaynes, E.T.: Probability Theory: The Logic of Science. Cambridge University Press (June 2003)
26. Diard, J., Bessière, P., Mazer, E.: A Survey of Probabilistic Models Using the Bayesian Programming Methodology as a Unifying Framework. In: Conference on Computational Intelligence, Robotics and Autonomous Systems, CIRAS. (2003)
27. Lebeltel, O., Bessière, P., Diard, J., Mazer, E.: Bayesian Robot Programming. Autonomous Robots **16**(1) (2004) 49–79
28. Bessière, P., Laugier, C., Siegwart, R.: Probabilistic Reasoning and Decision Making in Sensory-Motor Systems. 1st edn. Springer Publishing Company, Incorporated (2008)
29. Colas, F., Diard, J., Bessière, P.: Common Bayesian Models for Common Cognitive Issues. Acta Biotheoretica **58** (2010) 191–216
30. Le Hy, R., Arrigoni, A., Bessière, P., Lebeltel, O.: Teaching Bayesian behaviours to video game characters. Robotics and Autonomous Systems **47** (2004) 177–185
31. Synnaeve, G., Bessière, P.: Bayesian Modeling of a Human MMORPG Player. In: 30th international workshop on Bayesian Inference and Maximum Entropy, Chamonix, France (July 2010)