QOMPLX:

A Reasoning-based Defogger for Opponent Army Composition Inference under Partial Observability

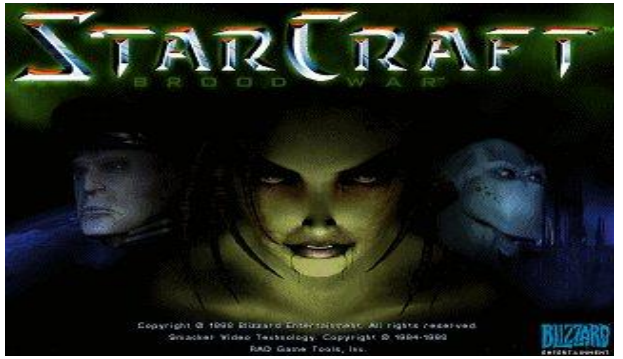
Hao Pan

QOMPLX, Inc.

10-19-2020

StarCraft and the Fog of War

- StarCraft: Brood War (SC:BW) is the expansion pack for the military science fiction real time strategy (RTS) video game StarCraft
- Why StarCraft? Challenging properties of real-world scenarios such as:
  - 512 x 512 2D map of “walkable tiles”
  - 200 units per player
  - 3 Factions and 45 unit types
- Fog of War (FoW) is the game mechanic where a player's access to information is limited to the area which is currently revealed by his/her own units, buildings, and abilities
- When a player loses sight of an area, either by a unit moving away or dying, the map area reverts to a dark shade and any information about units is removed from the main screen and the minimap
- The core challenge brought by the FoW: uncertainty. More specifically:
  - Natural environment
  - Uncertainty of the enemy and their intent
  - Communication among friendly units and decision making



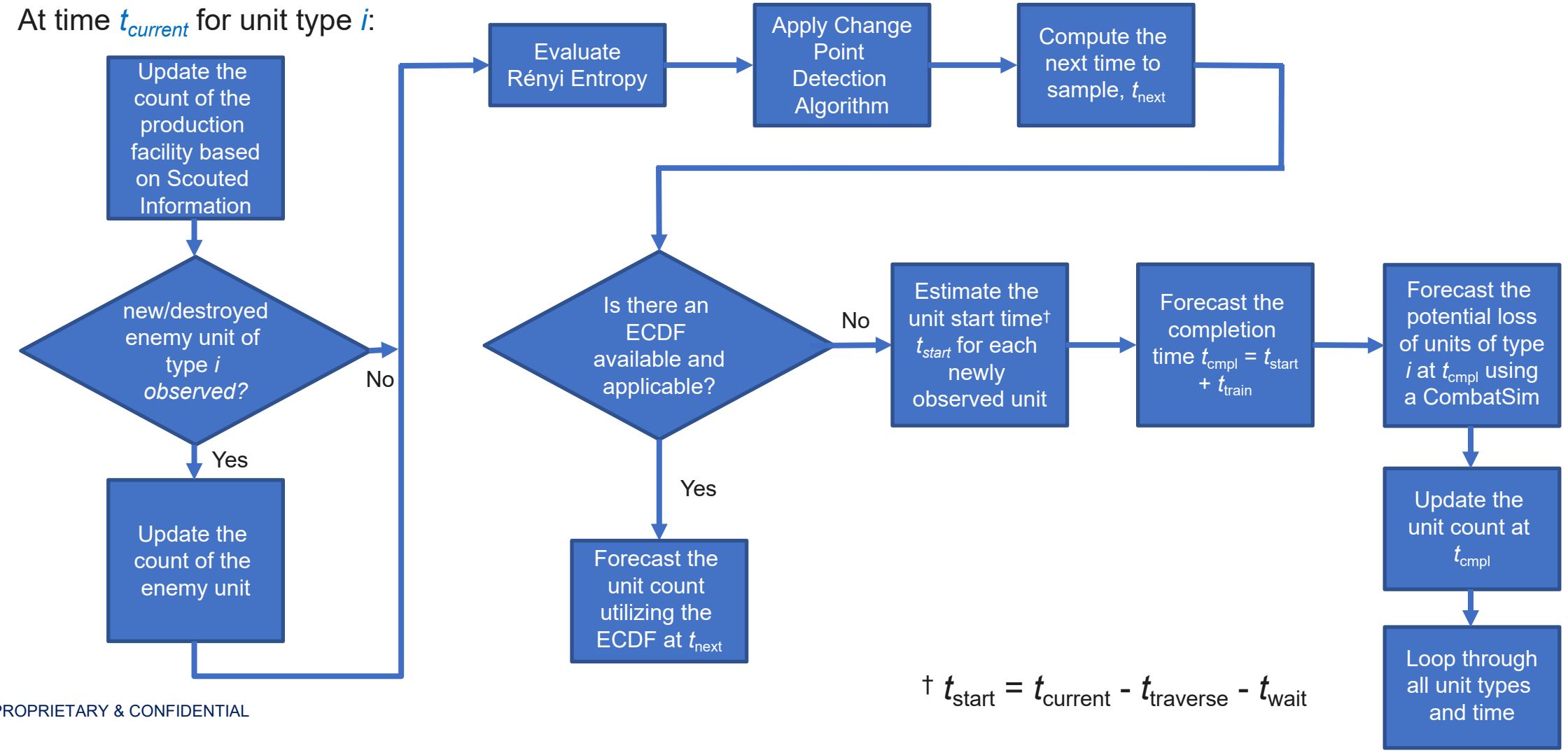
StarCraft: Brood War

Probe exploring areas covered by FoW

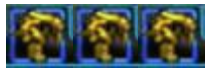
The Problem and Proposed Solution

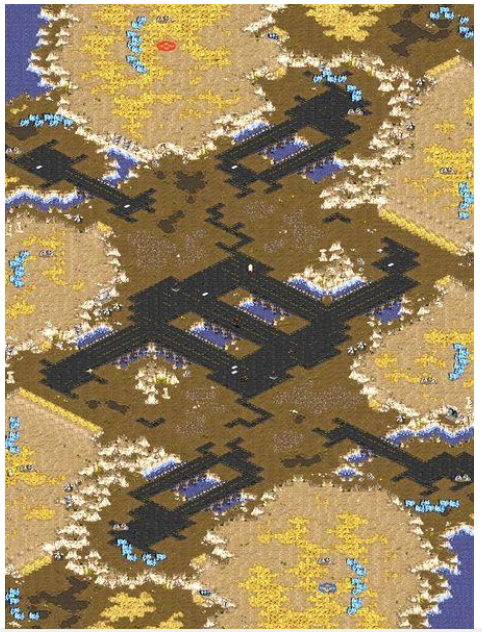
- We deal with the problem of inferring the opponent's army composition under partial observability
- More specifically, we are interested in:
  - What unit types the opponent has
  - How many units of a certain type the opponent has
- This whole process is what we refer to as defogging
- The core idea behind our proposed solution is to estimate the number of production facilities the opponent has
- Then we reason on how many army units the opponent will produce in the near future and of which type.

The Process of Dynamic Inferring



Case Study I

- The purpose here is to demonstrate how the proposed solution works
- We pitch our own bot “Hao Pan” vs the winner of the 2017/18 student devision: WuliBot
- WuliBot does a Zealot rush build exclusively 
- 10 Games were run on a single map: Destination, to minimize the influence of map factors



The map “Destination”

BASIL Ladder — About | Build Orders | Crosstable | Last 24h | Ranking | Files | Rules

Ranking

The following table is sorted by ELO, % Win, # Won by default. To sort multiple columns simulatenously, hold *shift* when clicking a second, third... column.

#	Bot	ELO	# Games	# Won	# Lost	% Win	Δ Crashes	Last updated
1	krasi0	2895	19205	17660	1199	93.64%	0	2020.01.13 08:37 am
2	PurpleWave	2869	17797	15372	2076	88.10%	0	2020.03.25 01:37 pm
3	Locutus	2849	19873	17865	1488	92.31%	0	2020.03.31 09:41 am
4	Hao Pan	2740	19292	14512	4343	76.97%	1	2020.04.06 01:43 am
5	adias	2737	7094	6292	776	89.02%	1	2019.11.12 09:38 am

SSCAIT 2017/18

The 2017/17 installment of SSCAIT sported 78 participants. The tournament was divided into two divisions:

- **Student division**: Round Robin tournament of 6006 games, where everybody played two games against everybody else.
- **Mixed division**: Double elimination bracket of 16 best bots (including non-students and teams).

Student division results:

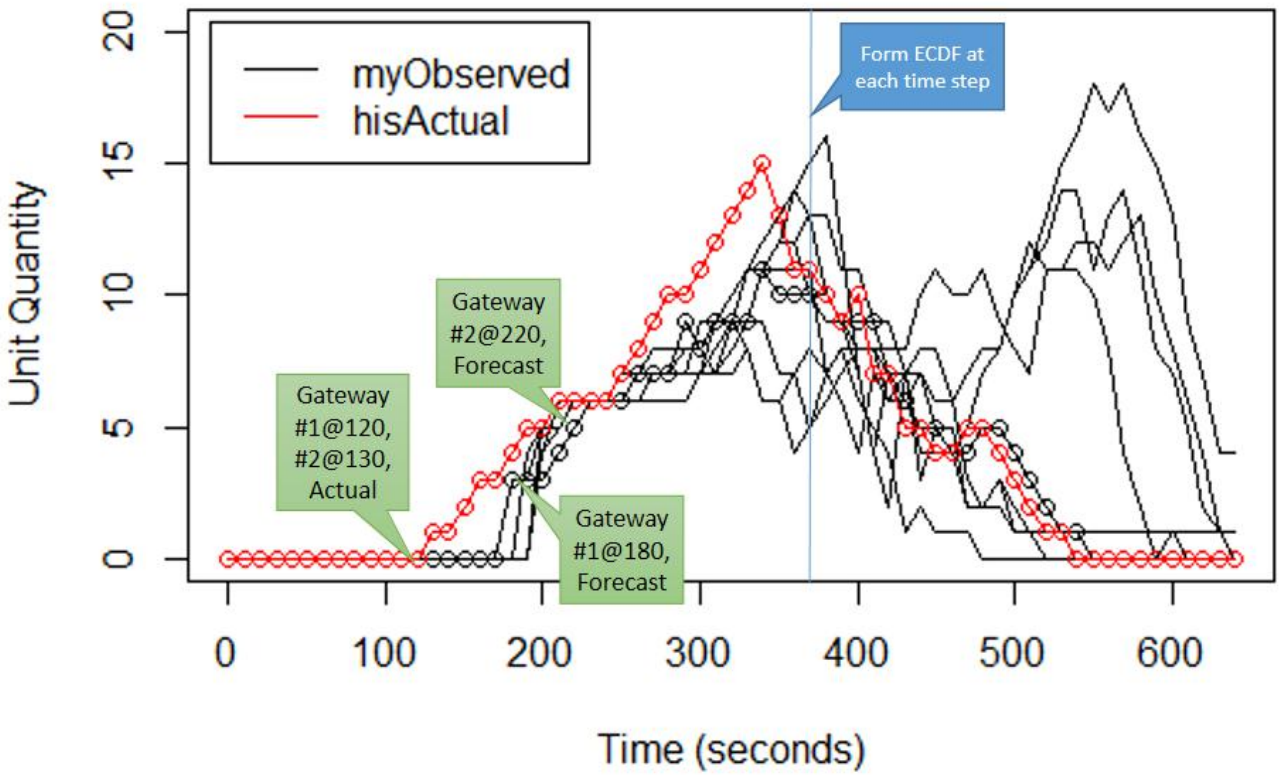
- 1st, 124pts: **WuliBot**, University of Southern California (USA)
- 2nd, 109pts: **Martin Rooijackers**, University of Maastricht (Netherlands)
- 3rd, 101pts: **Carsten Nielsen**, Technical University of Denmark (Denmark)

Mixed division videos:

The Mixed division bracket winner is **CherryPi** bot, made by **Facebook** Research team.

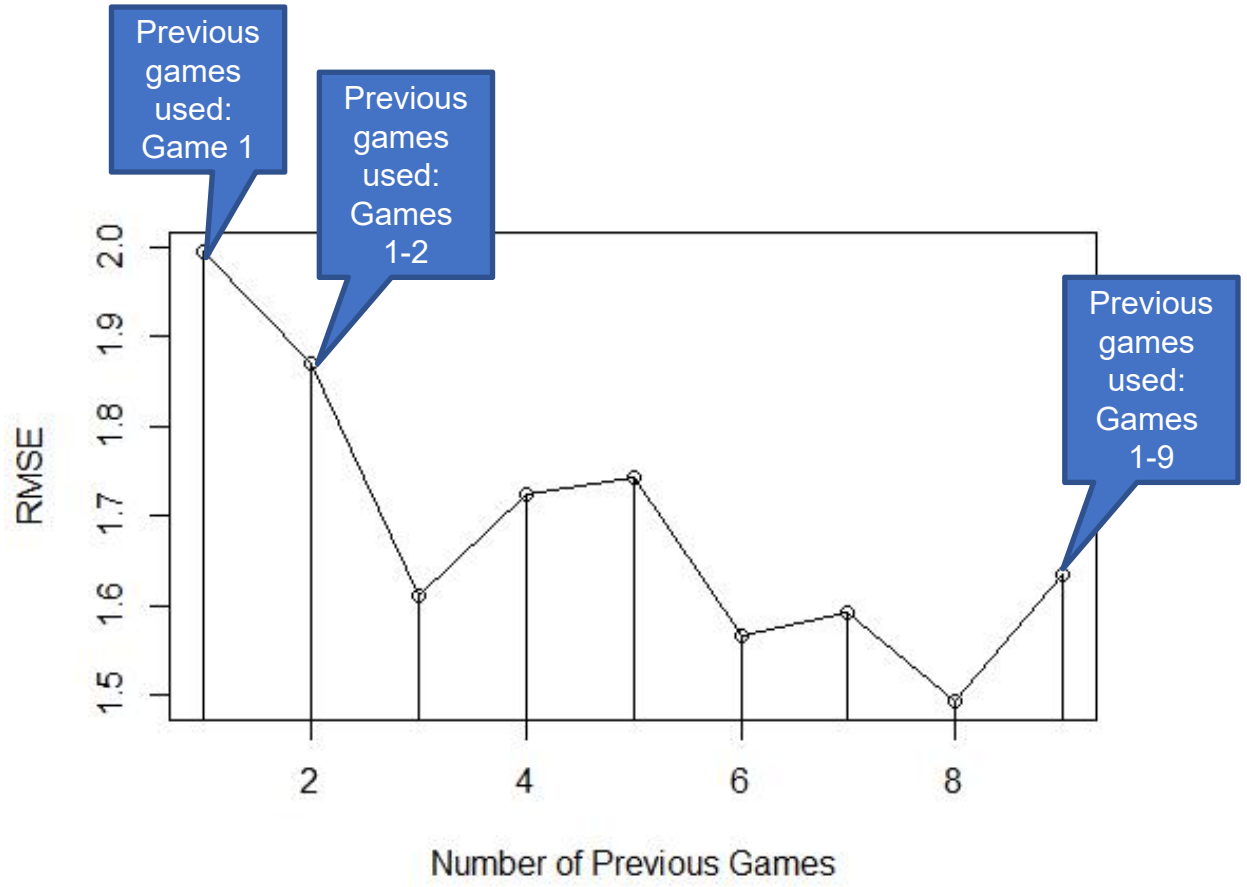
All the mixed elimination games can be watched in this YouTube playlist.

Case Study I, the Core Element



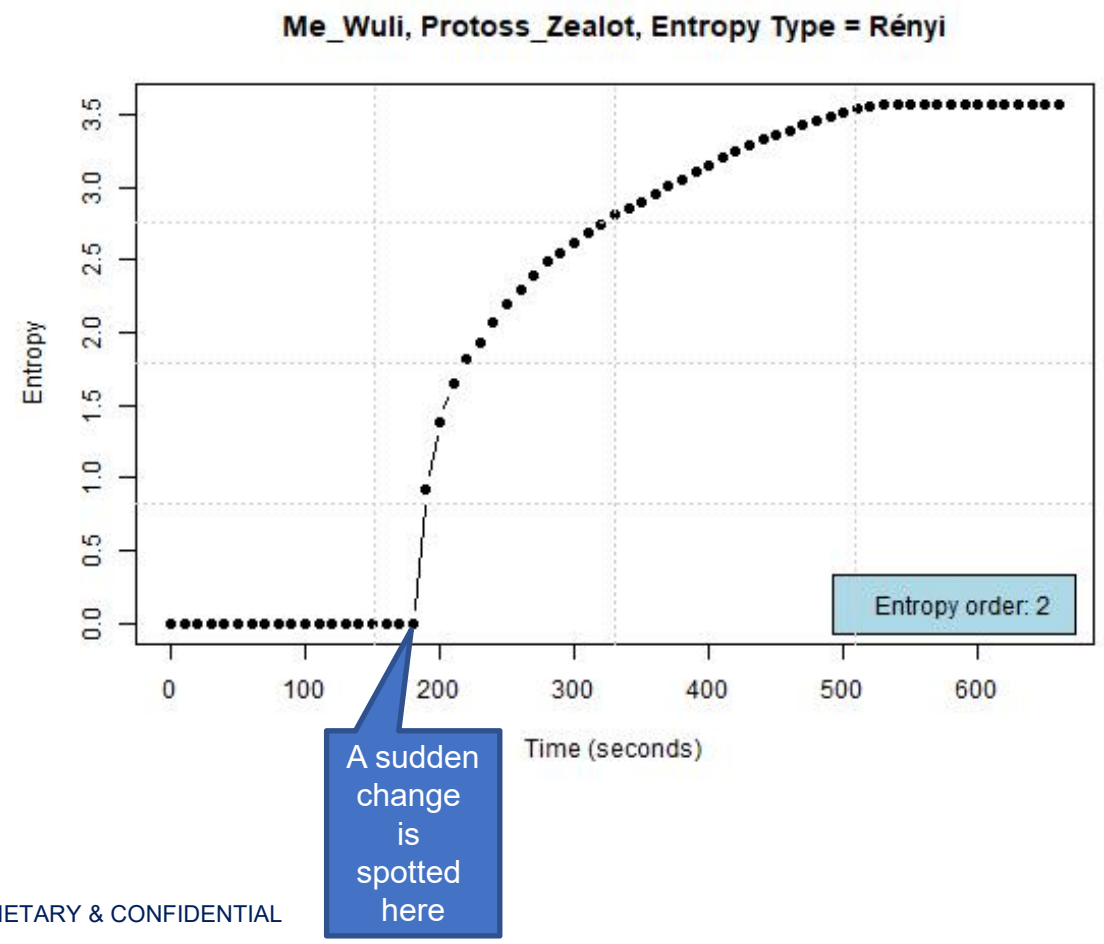
- We use the first 9 games as the basis to build the ECDF
- We use the 10th game to perform the defogging
- The opponent's first production facility, Protoss_Gateway, was forecast at time point 180 based on the first Protoss_Zealot spotted
- The second Protoss_Gateway was inferred at time point 220 as 2 more Protoss_Zealots were spotted
- Both pieces of information were confirmed by our scouted information
- The ECDF between time points 500 and 650 is to be disregarded due to the excessively large variance there

Case Study I, Previous Games & ECDF



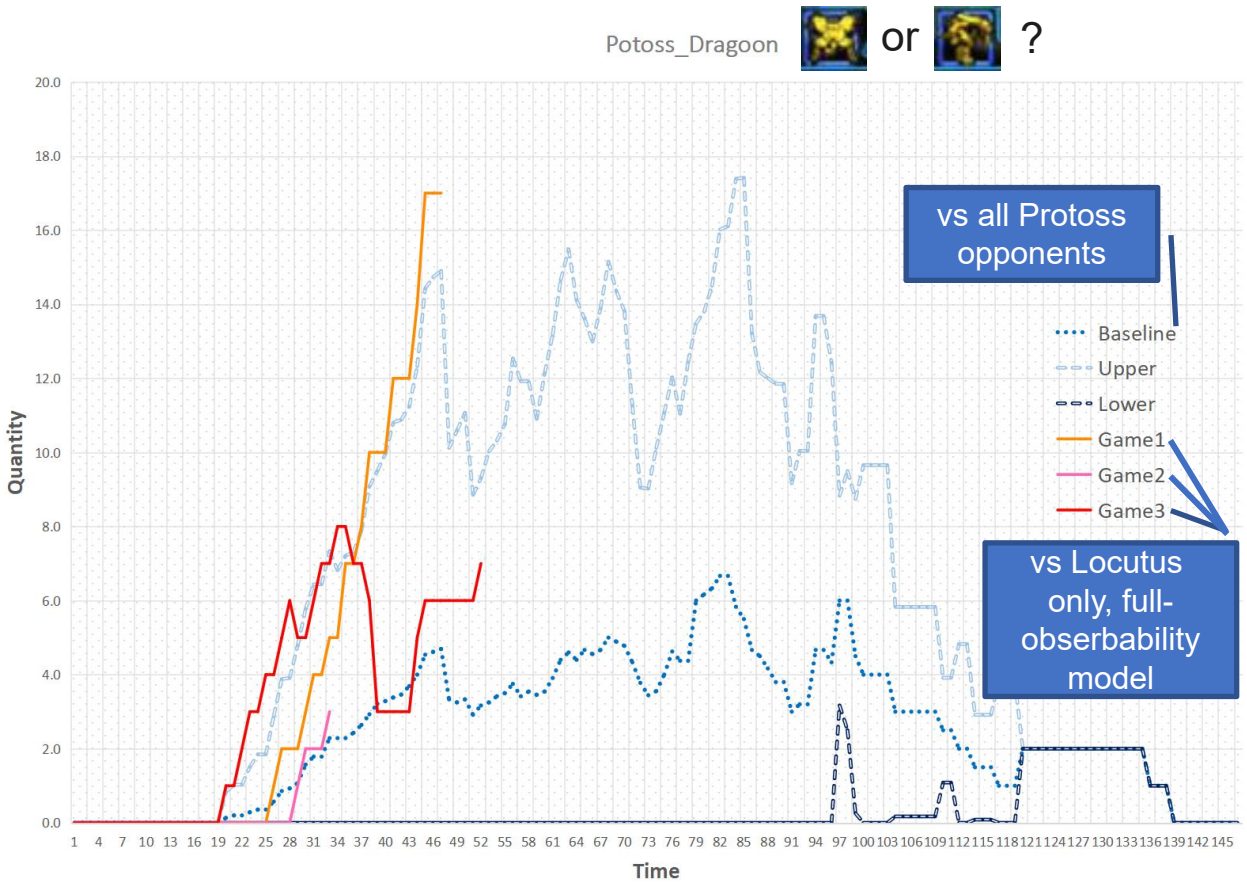
- We examine how previous games affect the defogging
- We evaluate the RMSE as the number of previous games used to built the ECDF increases
- The trend is generally downward, indicating the defogging gets better as the ECDF captures more information
- The number of previous games is low (9 at maximum), the use of ECDF is still justified due to its property of being able to work with low amount of data

Case Study I, Additional Notes



- We utilize the Rényi entropy coupled with a change point detection algorithm to determine the sampling rate
- At time point 180, the change point detection algorithm identified a sudden change in the entropy value
- This is confirmed by the fact that we did observe a sudden increase in the Protoss_Zealot count at that time
- We then increase the sampling rate to capture the subsequent infromation more thoroughly

Case Study II



- The purpose here is to compare the proposed solution with a few mainstream approaches
- We also set up both a baseline model and a full-observability one to see how effective an approach is at handling the FoW
- We also chose a much more challenging opponent: Locutus which has multiple build orders and is adaptive in its army compostion

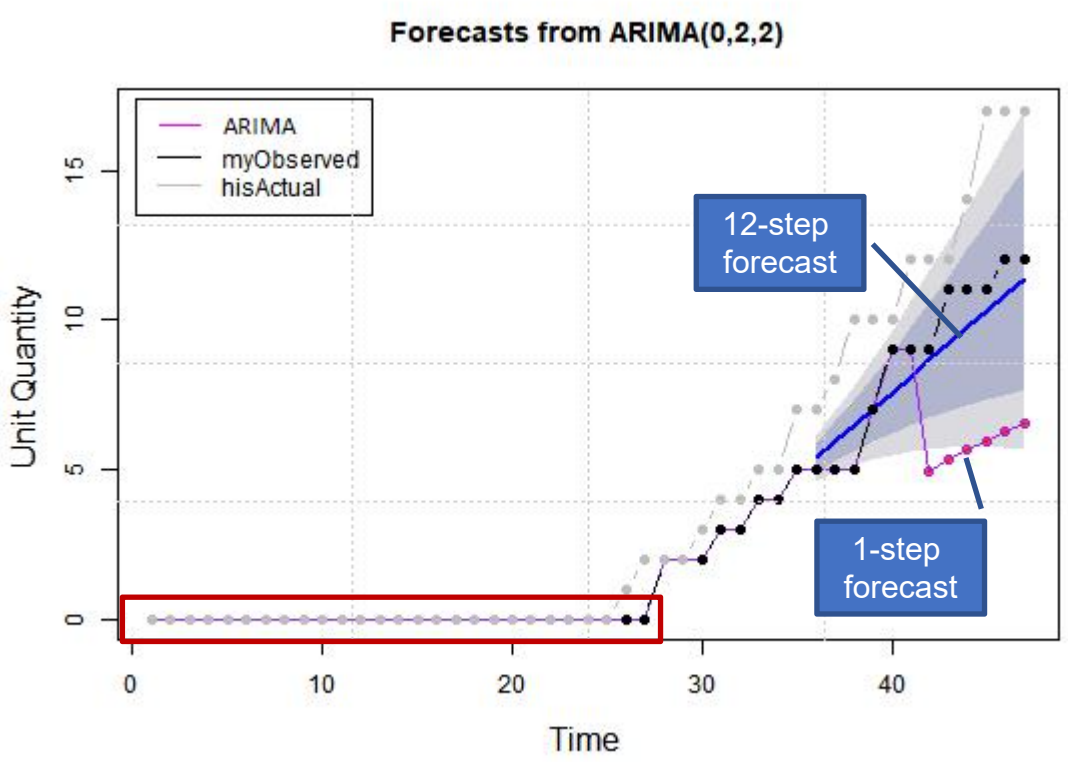
BASIL Ladder — About | Build Orders | Cross

Ranking

The following table is sorted by ELO, % Win, # Won by default. To sort multiple columns simulatenously, hold *shift* when clicking a seco

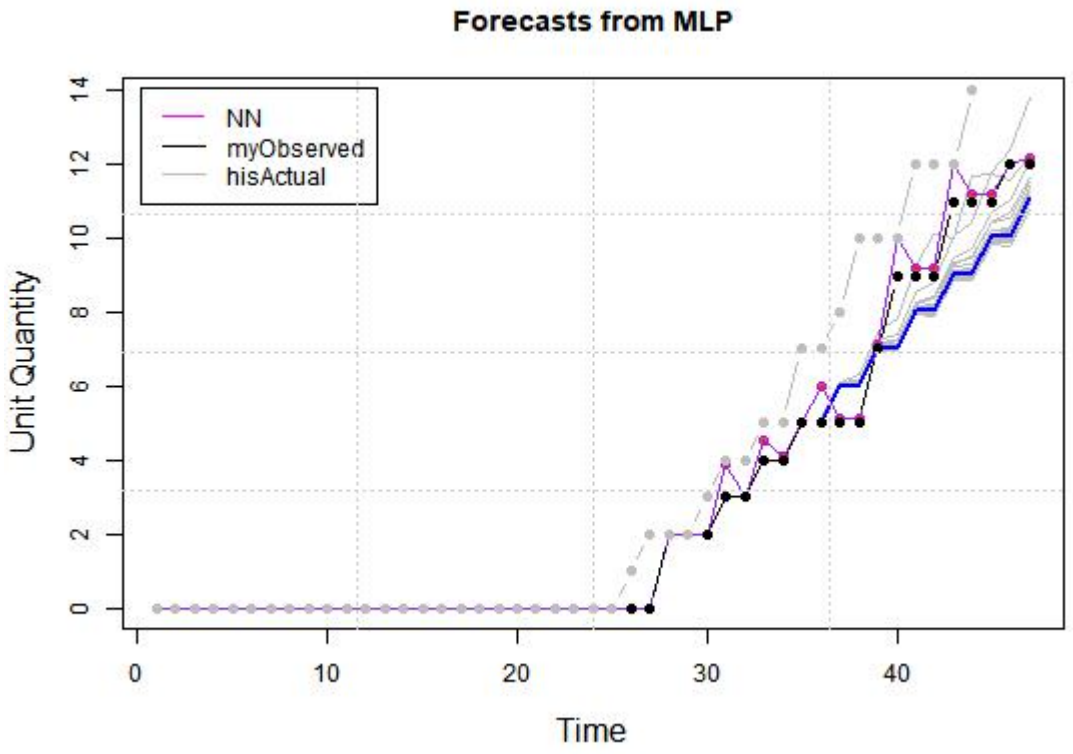
#	Bot	ELO	# Games	# Won	# Lost	% Win
1	krasi0	2895	19205	17660	1199	93.64%
2	PurpleWave	2869	17797	15372	2076	88.10%
3	Locutus	2849	19873	17865	1488	92.31%
4	Hao Pan	2740	19292	14512	4343	76.97%
5	adias	2737	7094	6292	776	89.02%

Case Study II, ARIMA Model



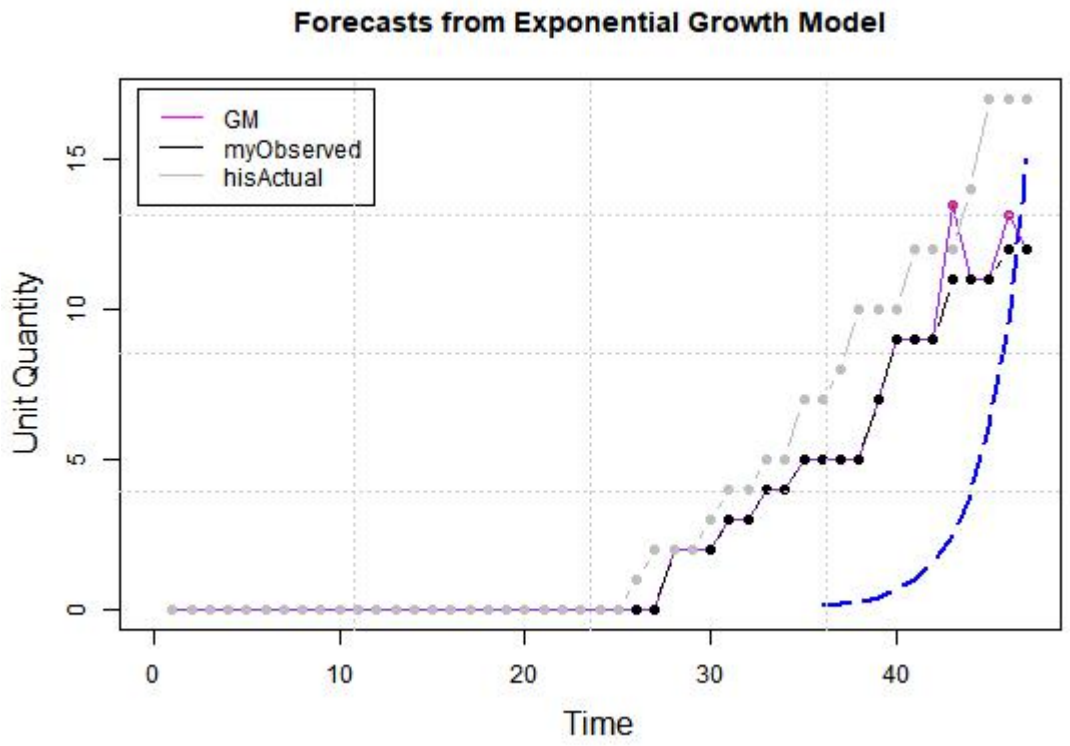
- The orders of the AR and MA parts, and the number of differencing are chosen automatically
- The ARIMA (0, 2, 2) model fitted here is effectively applying a linear exponential smoothing
- The initial period which contains zero values “dragged down” the forecast values later on since the smoothing is not sensitive to new trends
- For the proposed solution: the updated information from either the scouted or real-time intelligence prevents us from being stranded by the past data

Case Study II, Neural Network



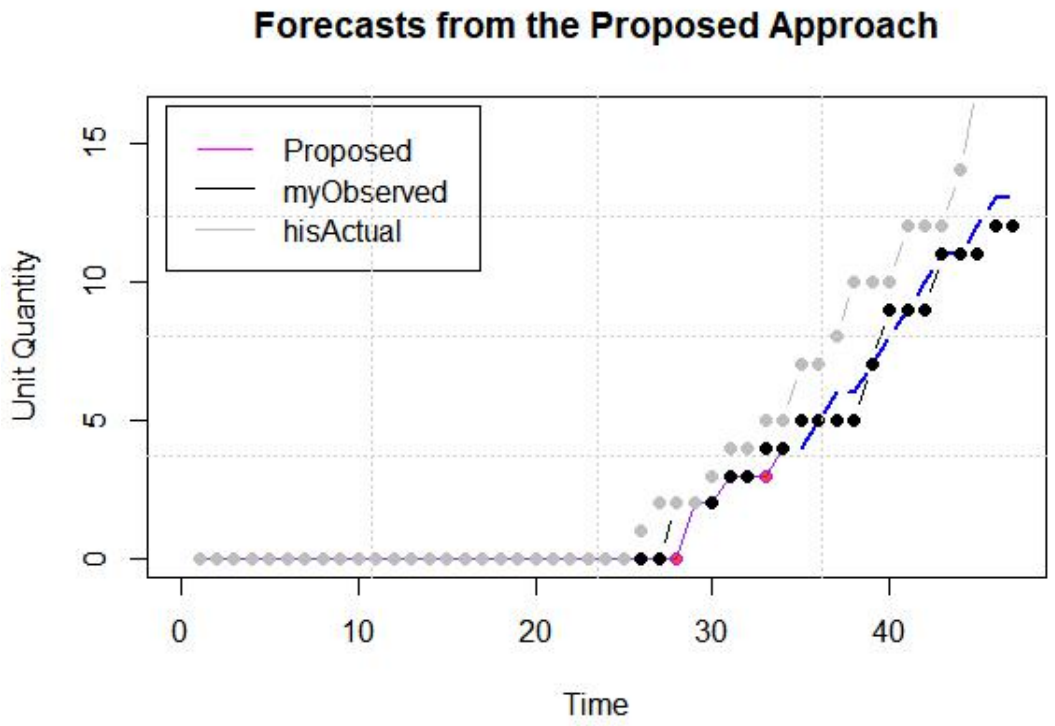
- We use a feed forward neural network called Multi-Layer Perceptron (MLP)
- Requires a mandatory burn-in period: at least10 data points were needed for MLP to produce meaningful forecasts
- There’s the issue of hyperparameter tuning where one needs to decide the number of layers and the number of nodes
- The computation cost is high
- One advantage is that it would work for any game, whereas the proposed method might have to be redesigned for a different RTS game
- For the proposed solution, there's no hyper-parameter tuning and the computation cost is minimal

Case Study II, Growth Model



- We use the the exponential type as the logistic type is quite sensitive to the zero values in the beginning, resulting in unsuccessful fit
- Also, the logistic model assumes that the growth rate decreases with population abundance, which is not necessarily true in StarCraft
- The predicted values are unreasonably huge towards the end and the fit is poor overall
- The model does not consider limiting factors such as available supply and resources
- For the proposed solution, it is relatively model free

Case Study II, Proposed Solution



- We correctly captured the growth rate of the Protoss Dragoon unit
- The slope of our forecast is roughly matching the that of true values, thanks to the correct inference of the number of production facilities the opponent had
- The combat simulator we use also contributed greatly as the casualty of the Protoss Dragoon unit can be significant at times
- There's a slight delay in our forecast, which needs further investigation

Case Study II, RMSE Values

		ARIMA	MLP	Growth Model	Proposed	Baseline
Game 1	1-step	3.5	1.8	1.9	1.7	4.7
	12-step	4.1	4.3	9.1	3.1	9.0
Game 2	1-step	2.1	0.8	0.9	1.0	2.9
	12-step	1.3	1.4	132.0	0.8	4.7
Game 3	1-step	2.4	1.7	1.7	1.2	2.6
	12-step	4.5	5.8	28639.0	1.1	2.1

- The proposed solution outperforms the rest in most cases
- Locutus went for a single unit type for a really long time before adding in other unit types into the army mix
- This is still challenging for a few approaches here
- The proposed solution handles this case well by taking advantage of the fact that a Protoss production facility can only produce one unit at a time
- In game 3 the casualty of Locutus' Protoss Dragoon unit was exceedingly high. The use of a combat simulator proved pivotal for our appraoch to produce a reasonable forecast

Conclusion and Future Work

- A reasoning-based defogger for inferring opponent army composition was proposed to address the challenges in RTS games with partial observability
- We utilize ECDF to handle the uncertainty brought about by the Fog of War
- We also employ information entropy to adjust the sampling rate properly
- The proposed framework was demonstrated to be superior than a few other common/popular solutions
- What we look forward to:
  1. Perform proper opponent build order classification/identification
  2. Improve the inference on the opponent intent
  3. Reducing the delay in the forecast further

Acknowledgements

- The authors would like to thank Nathan Roth (MSc) for his effort in helping creating our own bot for testing purposes in this research
- The authors would also like to thank Dan Gant from Facebook AI Research for his technical review and constructive comments
- The authors are grateful for Dennis Waldherr and his BASIL ladder on which testing and replay analysis can be done in a timely fashion

Thank You!

- How to reach me: hao.pan@QOMPLX.com
- My main research areas are:
  - ❖ Build order optimization
  - ❖ Dynamic adaptation of units/structure/upgrades/technology
  - ❖ Path finding/army movement