Team Naova

RoboCup Standard Platform League
Team Profile Paper 2019

Jonathan Fortin,
Thierry Pouplier,
Vincent Girardeau,
Ariane Beaudry-Betournay,
Antony Brochu

Naova Lab, École de technologie Supérieur (ÉTS), Montréal, QC, Canada
https://clubnaova.ca/

January, 2019

1. Introduction

Last year's Montréal hosted 2018 Robocup competition, was Naova's first time at participating to this event. Composed of 10 engineering students from École de Technologie Supérieure (ÉTS) - Montréal, Québec, Canada. The team leadership is composed of the following ETS students: Ariane Beaudry-Betournay (team captain), Jonathan Fortin, and Thierry Pouplier. ÉTS is recognised as the 2nd largest engineering university in Canada. ETS has provided the Noava team with dedicated facilities (e.g. private classroom) equipped with all the necessary hardware, software and robotics technologies needed for this project. The team was able to secure a number of partners, although, apart from the university, many of these partnerships are dependent on the team's successful qualification.

Naova is a relatively young team, with one year experience in the standard platform league (SPL) division and in various other divisions in robotics, and we strive to develop our skills in the SPL division. As first timers, at the RoboCup 2018 event, the Noava team successfully reached the finals of the challenger division and brought home the second place title.

2. General information

2.1. Team profile

Composed entirely of ETS students, members are offered research / class credits through Naova, however a number of team members merely list it as extracurricular activity. The team consists of a pool of a diversified and talented group of engineering students with various background expertise whom all share the same passion for robotics and its applications.

The 2019 Naova active team membership is composed of the following: Bétournay-Beaudry Ariane (team captain), Fortin Jonathan, Pouplier Thierry, Girardeau Vincent, Brulard Benjamin, Lussier, Marc-Antoine, Brochu Anthony, Zelaya-Diaz Edwin, Lussier Etienne, Caron Benjamin, Larbi Berkane-Hicheme, Prud'Homme Simon, Duchesne Florent, Lemire Laurent.

2.2. Positive impacts of last year's 2018 RoboCup

As a result of the successful first time participation at the 2018 RoboCup competition, the Naova team will be provided (May 2019) new facilities, adapted to accommodate the team's growth requirements (e.g. membership and space) including a dedicated practice field. Given the increased visibility and the demonstrated rapid maturity and progress of this young robotics team, combined with the planned participation at the upcoming Australian 2019 RoboCup, Noava has seen significant financial and scope impact for current and future innovation projects.

Last year's RoboCup competition brought home insightful lessons learned, while the second place significantly fueled the team's determination to further optimize game strategy in addition at bringing its very first impactful innovation to the RoboCup competition.

The Naova team has been collaborating with key ETS research groups including;

- « Le Groupe de recherche en électronique de puissance et commande industrielle » (GREPCI[1]) - specializing in industrial robotics (since 2017)
- The control and robotics laboratory (CoRo[2]) focusing in collaborative industrial robotics – (since 2019)

The goal of these collaboration efforts, is to further optimize Naova's work while leveraging their research as a baseline to Naova's future innovation in the field of the walking engine and the stability of the Nao.

2.3. Code Framework

The Naova code base uses the B-Human framework. The 2018-2019 focus will be to consolidate (migrate) all code artifacts onto the B-Human solution framework. It should be noted that the behavior, communication, special actions and the range of movement will either be modified, replaced or optimized by the innovation features presented in the following sections.

The team's key objective this year is in consolidating/eliminating any outside code artifacts while focusing the majority of innovation and creative ideas to remediate major weaknesses (as described in section 2.4).

2.4. NaovaCodeRelease2018 remediation scope

- Elimination of compatibility issues between the B-Human framework and Ubuntu 2018
- Remediation of communication issues caused by stack overflow in the output message of the Nao (e.g. improved validation of pre-conditions as well as message preprocessing logic used build messages before they are sent)
- Improved mobility features:
  - Arms behind the back: One of the challenges of our 2018 participation was the significant number of penalties encountered. To help with this, we have added the ability for our robot to put its arms behind the back. This strategy has proven to be successful by a number of teams, while reducing the number of falls and improved game fluidity.
  - Head movement: Effective ball location was one of the biggest issues encountered in last year's RoboCup competition. The enhanced head movement will significantly improve the accuracy to find the ball on the field while optimizing energy consumption by eliminating unnecessary full body rotation with more efficient head movement.

[1] GREPCI: https://en.etsmtl.ca/Unites-de-recherche/GREPCI/Accueil
[2] CoRo: https://en.etsmtl.ca/Unites-de-recherche/CoRo/Accueil?lang=en-CA

- Optimized logic surrounding role changes:
    - Effective role changing proved to be one of our biggest success at the RoboCup 2018. During the competition, the code release improvement contained the required logic behind the smooth role changing between the "stricker" and the "defender" when the Nao came back from a penalty.
    - Prior to the code fix, role changing anomalies were caused by a lack of verification of who was the nearest to the opponent penalty zone with the ball or closer to the ball and the opponent penalty zone than the player that came back to the field.

3. Behavior Control

Another lesson learned from the 2018 RoboCup competition, was around the need for improved decision process flexibility enabled by the implementation of new levels of abstraction in the state machine as well as the addition of a configuration graphical user interface (GUI) including machine learning features, described below.

3.1. Tactics management

One of the new state machine level is the "tactics". The purpose of tactics is to describe and organize the Nao team decision making capabilities. This includes improved role choice and actionable team intermediate goals.

3.2. Role management

Roles are used to establish the individual strategy while unlocking the list of possible actions that a Nao player can use to autonomously achieve its objectives. Additionally, roles provide an optimized structure for the strategic cooperation between all five Nao players on the field.

The Nao field players also have the ability to autonomously reconfigure themselves (via role switching) depending on the chosen strategy or when one of the players is retired from the field. This way, the player structure dynamically adjust itself while ensuring that the most critical positions are fulfilled at any given time.

It should be noted that significant code refactoring, is still progressing, to optimize existing roles through improved GUI around the strategy management functionality (see Sec. 5).

3.3. Skills management

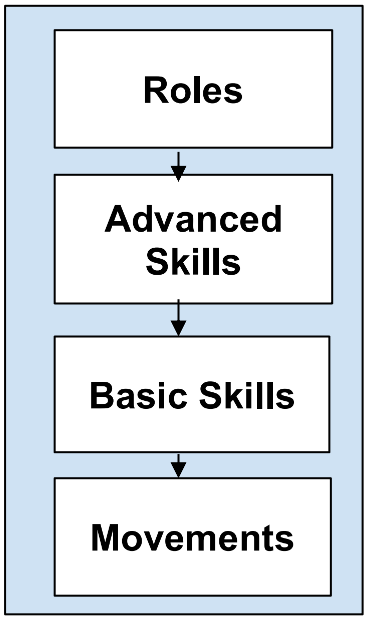


Figure 1

Skills management, is another key feature used to further optimize Nao capabilities. Through the addition of machine learning features used by the decision process algorithms, it soon became apparent the need to implement an action repository. These action repositories were then leveraged to elaborate various strategies as well as refining role behaviors.

“Skill” logic flows as described in figure 1, were further categorized (refined) to facilitate code encapsulation (abstraction) while avoiding direct granular action calls, consequently eliminating the risk of unintended mobility anomalies while standardizing object modularity and skills behavior.

3.1.1. Advanced Skills

Advanced skills are known as the orchestration of complex actions, which are used in specific circumstances in the course of a “game”. These advanced skills are essentially the sequencing of basic skills, deemed useful in a given game condition. Advanced skills never directly call granular actions to help preserve object abstraction and code encapsulation.

3.1.2. Basic Skills

Basic skills are known as the sequencing of granular actions leveraged by advanced skills.
Basic skills orchestrate the sequencing of directly called granular actions.

4. Motion management

Since last year, significant effort was put into building a comprehensive repository of available actions for our Nao robots. This ultimately allowed a better mastery of the soccer game itself. In addition, the rich action repository gave us improved flexibility while improving robot behavior control complexity.

4.1. Side kick (lateral kick)

Another identified weakness was around sidekicks (lack of). When two robots of opposing teams approach the ball while facing each other. The winner was often determined by luck, given equivalent starting distance and robot stability. To improve the predictability of a successful outcome of this game condition, we implemented a side kick. This enabled passing the ball quickly to a teammate on the side.

4.2. Pass

Another challenge was around pass strength (lack of) between the Striker and the Supporter. We decided to refactor and create a larger kick inventory to improve the accuracy and strength of the passing game.

4.3. Kick to the goal

Robot stability was also an identified deficiency, when attempting to score goals, originally our robots often lost balance (over 75% of all goals attempted). Since the RoboCup 2018 competition, significant improvements have been made around robot stability as well as kick strength and accuracy.

5. CoachApp Graphic User Interface (GUI)

The CoachApp GUI enables the easy management of robots and strategy used during a given soccer match.

5.1. Key features

Allows the coach to easily update (on demand) a number of robot configuration parameters to individual robot(s) and/or cascade adjustments to all active robots on the field regarding;

- Wifi config
- Field layout
- Match strategy
- Role switching
- Throttle action parameters (e.g. kick strength, kick direction such as side or front kick)
- Robot default field positioning in context of number of remaining players on the field, strengthen defense positions etc …)

5.2. GUI framework used

Frontend: JavaScript, HTML and Electron

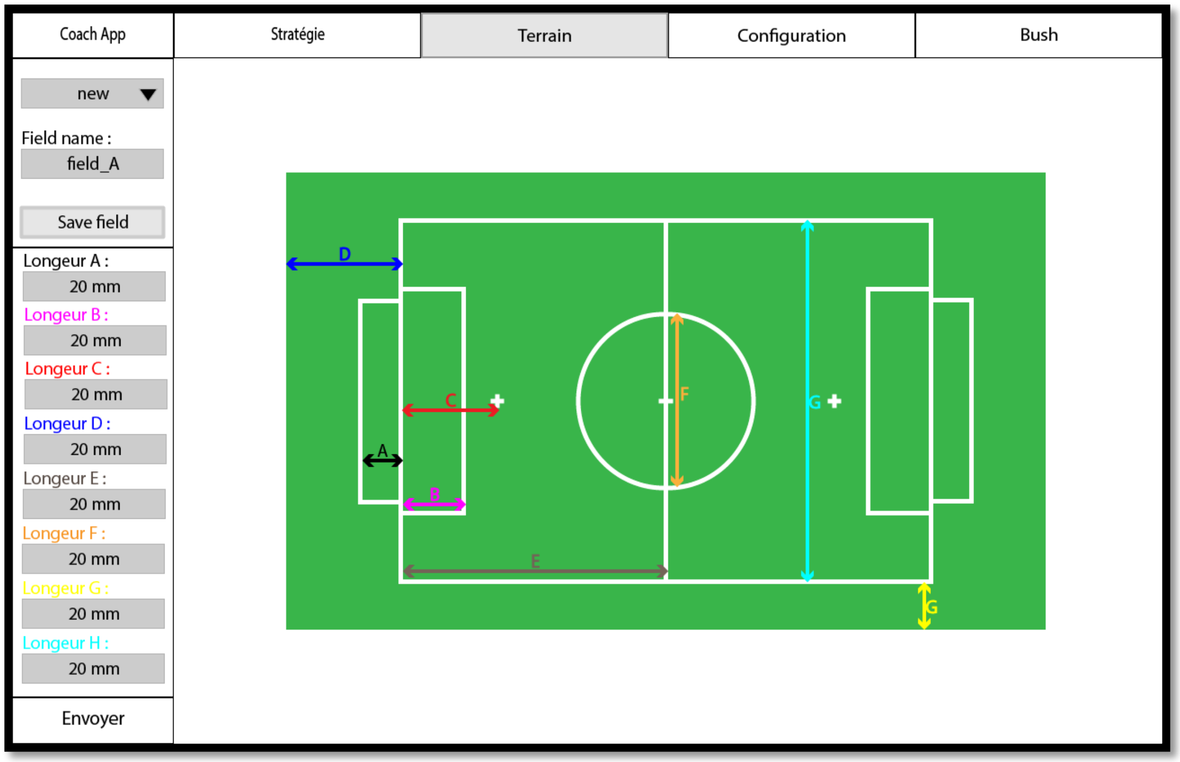
Design principles: Platform independent, interpreted, ease of use, well known frameworks not requiring specific skill sets.

5.3. Key GUI sections

5.3.1. Field management GUI

The field management GUI allows adjusting field dimensions for a number of different field layouts, while also allowing on demand adjustments regarding the current field used.

The GUI greatly simplifies field layout definitions and management, requiring only a handful of manually entered key measurements while remaining field characteristics are automatically calculated.



Field name : field_A

Longueur A : 20 mm

Longueur B : 20 mm

Longueur C : 20 mm

Longueur D : 20 mm

Longueur E : 20 mm

Longueur F : 20 mm

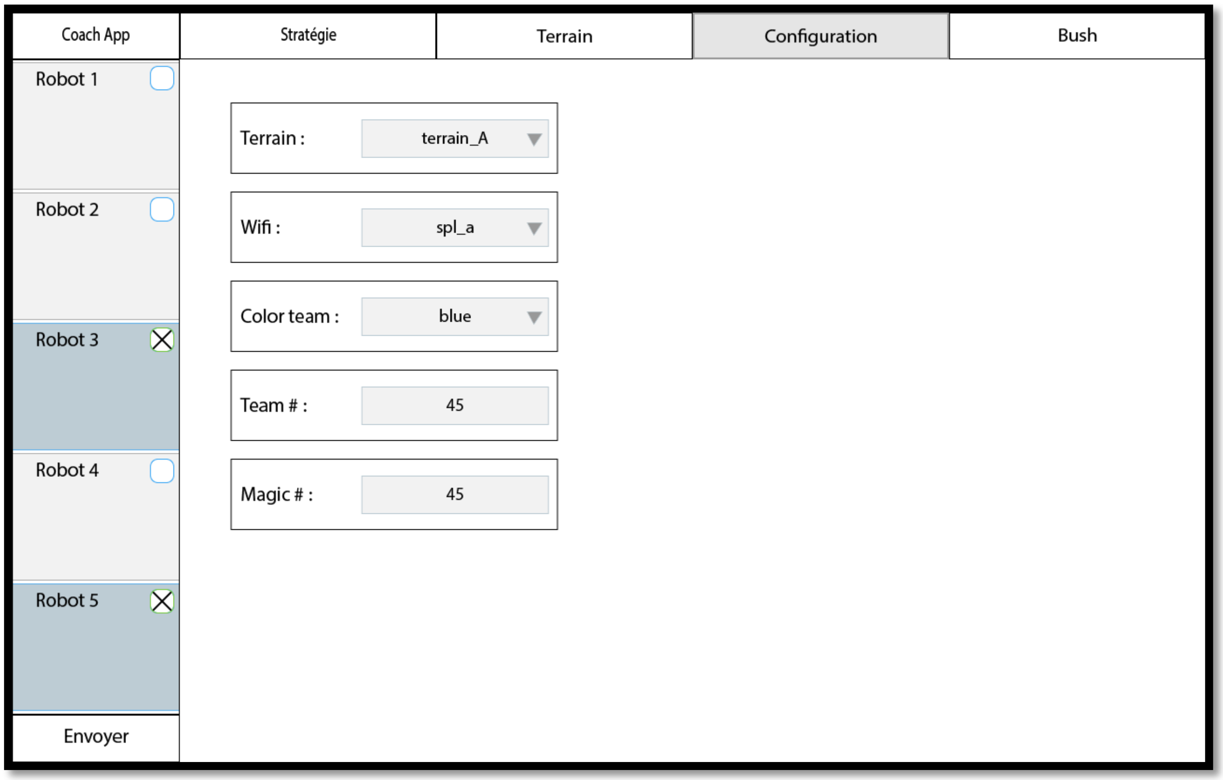
Longueur G : 20 mm

Longueur H : 20 mm

Figure 2

5.3.2. Robot configuration management GUI

This GUI enables the central management of Robot configuration features such as assigned Field ID (e.g. Terrain), Wifi, team color, team number and more. The GUI essentially sends all the configuration for the toolbar of the Bush module while avoiding code redeployment every time.



Terrain :	terrain_A
Wifi :	spl_a
Color team :	blue
Team # :	45
Magic # :	45

Figure 3

5.3.3. Strategy management GUI (still in development)

The strategy management GUI is a key feature used by the Coach during a game, as it allows the on-demand management of game strategy to be used by all robots throughout the game.

Position sub GUI	This GUI section will allow optimal robot (player) field positioning, robot actions to be enabled (or disabled), adjust specific robot field positioning as the game unfolds (e.g. including dynamically establishing defensive player wall and/or strategically spreading players across the field in either a defense or offensive role as needed).
Role sub GUI	This GUI section will allow on demand role creation and robot association to those roles, while allowing dynamic switching of roles (usually when penalties are called) among in field robots.
Action sub GUI	This GUI section will allow the dynamic management of skills associated to roles.

5.4. Explored solution options to dynamically convey config changes to field robots

5.4.1. Option 1: Dynamic generation of a state machine file

Consists of dynamically generating and communicating a state machine file to field robots;

Pros: This removes the complexity of the robot having to read a config file and rebuilding the associated state machine.

Cons: Likely cpu and memory intensive, as it would require robots to recompile code compounded by algorithm complexity of the actual application.

5.4.2. Option 2: Dynamic generation of a configuration file (preferred option)

Consists in communicating a multi-section config file (xml or equivalent), storing all required strategy information, to field robots. Anticipated Config file sections would be position, roles, and actions;

Pros: Simple and light weight, less cpu and memory intensive, does not require any code recompile

Cons: N/A

A robot algorithm would dynamically digest config changes as they come in.

6. Machine learning

The typical branches of machine learning are known as supervised, unsupervised and reinforcement learning.

Supervised	Supervised learning, consists of the data scientist who acts as a guide to teach the algorithm what conclusions it should come up with (also referred as discriminate input given the output during training)
Unsupervised	Unsupervised learning, consists of finding cluster in the data without been given any guidance. Unsupervised machine learning is more closely aligned with what some call true artificial intelligence — the idea that a computer can learn to identify complex processes and patterns without a human to provide guidance along the way
Reinforcement	Reinforcement learning, is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Although supervised and unsupervised techniques would be useful if used on the sensory input of the robot, it was decided to focus our research on reinforcement machine learning, which is better suited, for complex decision-making processes required during a soccer game.

6.1. Motivation

Simply put, playing soccer requires the ability to move efficiently, being able to quickly detect the changing circumstances of the game while being able to quickly make the most appropriate decision, ultimately leading to successfully scoring a goal.

The mobility and sensing features of our robots are relatively stable primarily thanks to the availability of open source code baselines. So, it was decided to focus the team's effort on optimized decision-making mechanisms.

Last year, after designing different tactics and roles for our Nao robots, it quickly became apparent the limitations of state machines. The overwhelming complexity quickly became unmanageable with every new functionality added. With the arrival of the Nao V.6, it was decided to switch to a combination of reinforcement learning for the roles, and state machine for the more predictable aspects of the game.

6.2. Languages and framework

Using the Bhuman stack as a baseline, Python to support the iterative development cycles, while the ultimate code is exported as C++ for the final generated models (for performance reasons).

Python is used to facilitate the iterative agile development software lifecycles (SDLC), primarily for its ease of use and rich machine learning libraries. The popularity of the language in the field as led the development of a number of comprehensive open-source frameworks extensively supported by the industry, enabling Nao programmers to operate at a higher level of abstraction. Python frameworks are particularly interesting for their ability to leverage C and Fortran libraries.

Lastly, the python code base (learning modules) are easily exported into C++ to ultimately meet the critical runtime performance criteria.

6.3. Model

The newly developed algorithms are expected to exceed last year's code in a match. Additionally, the algorithms will be able to better cope with the complexities and dynamic circumstances and constraints coming from the difficult task of playing soccer.

For Nao robot agents, adversaries are of course only one part of the environment, however the complexity is significantly compounded by a number of unknown unknowns, in terms of the ever-changing circumstances of a given soccer game (e.g. condition/state predicates, what other teams will do, what actions our robot agents will choose to be successful). Therefore, the environment must be considered as stochastic[3].

Although the choice of available actions a Nao robot can take is limited, the parameter range of those actions is continuous. This results in a relatively infinite action range. The same goes for the state machine space, as the field is not constituted of tiles. Considering that we ideally want a robot agent to be able to play for an entire game, this rules out any discretized algorithm [4]like dynamic programing or Q-learning [1] , as the resulting tables would too large for the Nao's limited SPL capabilities (e.g. hardware, cpu, memory). For that reason, Naova's research is focused toward the actor-critic model, as it has continuous state and action compatibility.

The robot agents should be able to efficiently work together. For instance, multiple same team robots should avoid going for the ball if one of them already has it. Instead, the remaining team robots should position themselves to successfully receive a pass or adopt appropriate defense positions depending on the assigned role for a given game strategy. To achieve this, robot agents are trained for each role. Each having their own custom reward function.

Also, the reward mechanisms the soccer game returns are relatively limited (e.g. successful scoring of a goal). A custom reward function should be implemented to ease the learning process. For instance, a context aware reward mechanism adapted to the individual robot role and/or game strategy should yield better results [2] (e.g. successful passing of a ball to a fellow robot team member, "goal keeper" successfully preventing the apposing team from scoring a goal, reward calculated from the distance between the ball and the enemy goal).

[3] Stochastic algorithms use randomness. They use all combinations but not in order but instead they use random ones from the whole range of possibilities hoping to hit the solution sooner. Implementation is fast easy and single iteration is also fast (constant time) Heuristics algorithms.

[4] Discretization algorithms have played an important role in data mining and knowledge discovery. They not only produce a concise summarization of continuous attributes to help the experts understand the data more easily, but also make learning more accurate and faster.

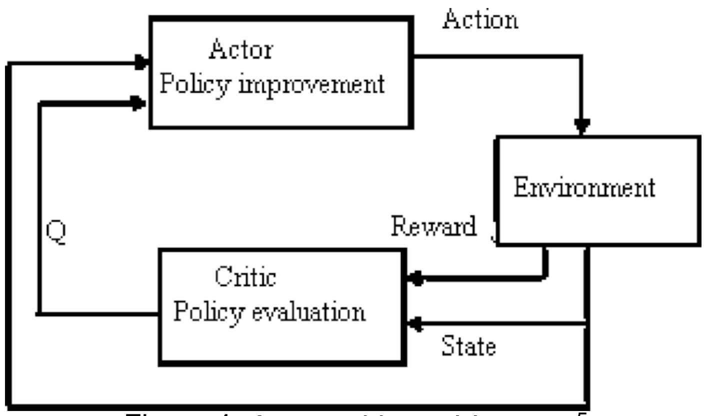


Figure 4: Actor-critic architecture[5]

A number of questions still need to be answered; For example, should we use a hierarchical architecture to obtain goals and sub goals to facilitate efficient machine learning [3] Which exploration mechanism to use to get the best result [4] Should the training be done with one robot agent at a time or try an asynchronous approach [5] Timeboxing decision making processes -The game has a preset time limit, therefore robot agents should be made aware of this time limit to influence (optimize) their individual decisions/actions toward the end of the game [6]These questions should hopefully be answered in the coming months as the machine learning project progresses. The plan is to have at least one operational model in time for the RoboCup 2019 competition.

7. Closing Statements

On top of the above mentioned inflight/completed 2018-2019 improvement objectives, the Naova team also plans to optimize “Goal keeper” strategies and motion, as another key weakness identified during the 2018 RoboCup competition.

It is clear that opposing teams will likely change their strategies (offensive/defensive) against Naova given they will no longer have the element of surprise of last year’s competition. However, the above mentioned improvements combined with optimized “Goal keeper” motion, Naova has a better chance to win the challenger division. The scope of the “Goal keeper” improvements seeks to achieve overall movement efficiency and stability with simpler and faster “Goal keeper’ motion. Also, the relationship between ball location mechanisms and the communication of the ball location between Nao players needs to be further improved.

Since last year’s 2018 RoboCup competition, Naova has significantly cleaned up, simplified and stabilized the baseline code base while remediating a number of identified weaknesses as well as introducing innovative machine learning mechanisms. Those changes will hopefully improve the overall predictability of Naova’s performance at the 2019 RoboCup.

Of course, this is a modest beginning for a young team ‘multiyear’ journey striving to position itself (Naova) as a formidable competitive RoboCup opponent.

[5] Figure 4 :https://www.researchgate.net/figure/Policy-iteration-and-actor-critic-learning_fig1_6170058

8. Bibliographie

[1] R. McFarlane, “A Survey of Exploration Strategies in Reinforcement Learning,” p. 10.

[2] V. R. Konda and J. N. Tsitsiklis, “Actor-Critic Algorithms,” in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 1008–1014.

[3] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning,” *ArXiv160201783 Cs*, Feb. 2016.

[4] Microsoft Research, *Deep Multiagent Reinforcement Learning for Partially Observable Parameterized Environments*. .

[5] A. Levy, R. Platt, and K. Saenko, “Hierarchical Actor-Critic,” *ArXiv171200948 Cs*, Dec. 2017.

[6] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, “Time Limits in Reinforcement Learning,” *ArXiv171200378 Cs*, Dec. 2017.