AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents

Vincent Conitzer & Tuomas Sandholm (2007)

JANNIK PETERSEN

13.06.2019

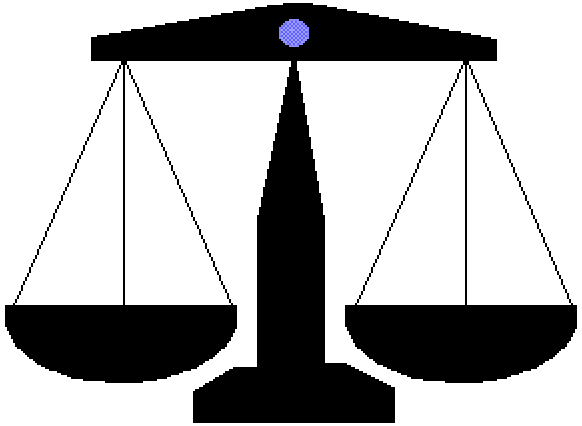
Overview

- Getting Started
  - Nash Equilibrium
  - Learning in Games
  - Setting
  - Play a Stage Game
  - Two essential Properties
- The Algorithm
  - AWESOME
  - Null Hypothesis
  - Why do we need equilibria?
  - Self-Awareness
  - What does AWESOME play
  - First Approach
  - Solution
  - The Algorithm
- Comparison
- Summary

Nash Equilibrium

- Describes a game situation
- Combination of strategies in non-cooperative games
  - Each player picks one strategy
  - For none player, it makes sense to change his strategy

- Pure strategies
  - Can react accordingly to stationary opponent
- Mixed strategies
  - Random choice of actions



[1]

Learning in Games

Aspects of learning a game:

- Learn the game itself
- Learn how the opponent is behaving

We focus on the second aspect:

- Assume that the game is known
- A equilibrium of the game can be computed

[6]

The Setting

- N players, each with their own set of actions
- The stage game is played repeatedly
- Agents choose their action independently

	1, 2	2, 1	3, 0
	2, 1	1, 2	4, 0
	0, 3	0, 4	2, 2

- For each epoch, the players decide to take a distribution of their set to play
- Players have a long-term strategy
  - Stationary strategy – play same distribution every time
  - Mixed strategy – play different distribution every time

Play a Stage Game

- Nash equilibrium:
  - Each player has a mixed strategy
  - It‘s a best response to the other strategies
  - Makes sense for rational players

	50%	50%	
50%	1, 2	2, 1	3, 0
50%	2, 1	1, 2	4, 0
	0, 3	0, 4	2, 2

- Problem: less clever opponent:

	49%	51%	
100%	1, 2	2, 1	3, 0
	2, 1	1, 2	4, 0
	0, 3	0, 4	2, 2

What do we want?

A satisfactory multiagent learning algorithm should learn to play optimally against stationary opponents and converge to a Nash equilibrium in self-play.

Two essential Properties

- Play optimally against (eventually) stationary opponents
  - -> Play best response
  - Maximum exploitation

- Convergence to Nash equilibrium in self-play

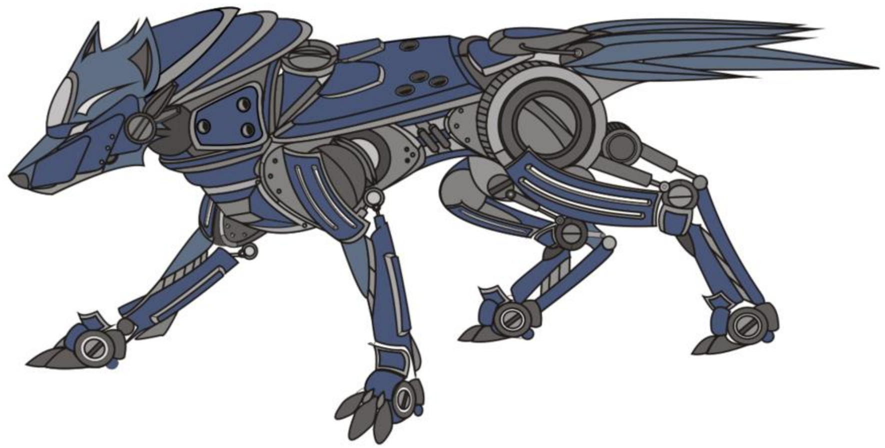
These properties are minimal!

REMOVED ASSUMTIONS

- Best previous example of a learning algorithm: WoLF-IGA

- WoLF-IGA : **W**in or **L**earn **F**ast – **I**nfinitisimal **G**radient **A**csent

- Assumtions:
  - There are at most 2 players
  - There are at most 2 actions per player
  - Each opponents strategy is observable
  - Gradient ascent of infinitisimal small step sizes



[5]

AWESOME

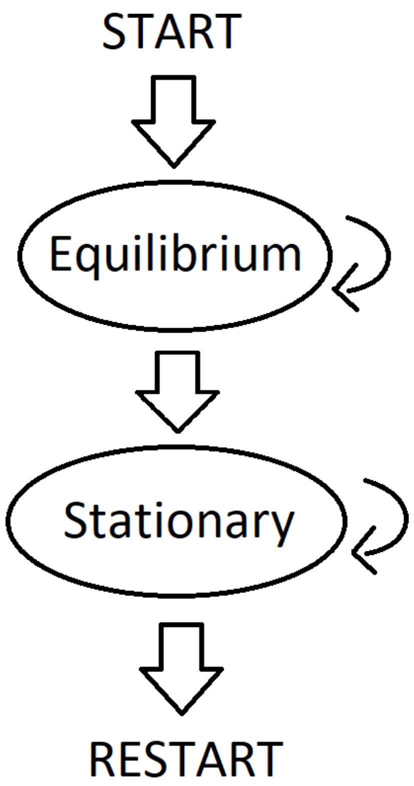
- AWESOME – **A**dapt **W**hen **E**verybody is **S**tationary, **O**therwise **M**ove to **E**quilibrium

- Basic idea:
  - Detect if opponent is playing stationary
    - Play best response
  - Otherwise, restart and go back to equilibrium

[4]

Null Hypothesis

- Start with first null hypothesis
  - Everyone is playing the precomputed equilibrium
- If this is rejected, switch to other null hypothesis
  - Everyone is playing stationary
- If this is rejected, restart completely
- Evaluate hypothesis every epoch



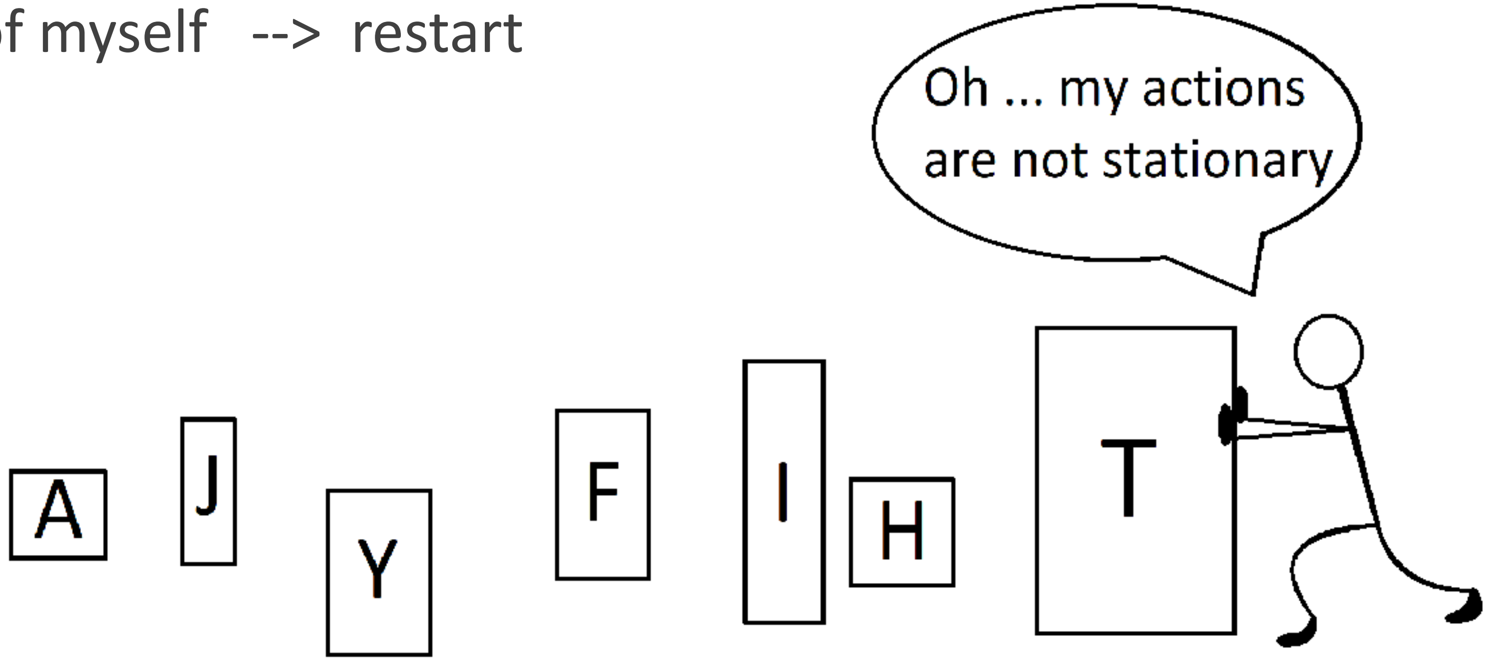
Why do we need Equilibria?

- In one-shot games
  - They are natural and simple
  - They always exists
  - They are robust to changes

- They are also in general repeated games

Self-Awareness

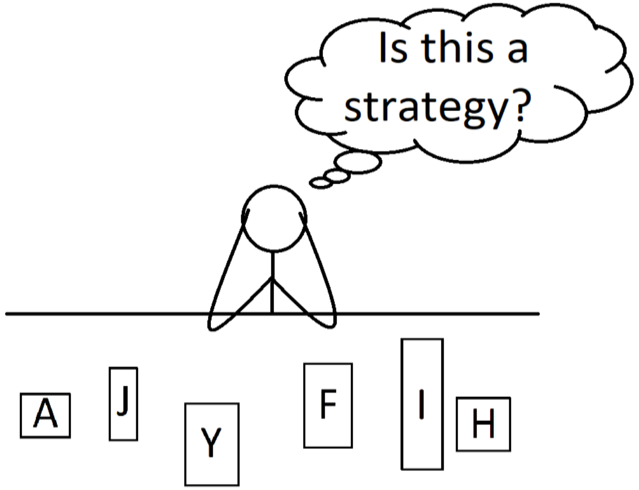
- AWESOME is self-aware
  - Detect nonstationarity of myself --> restart



What does AWESOME play?

- As long as Awesome accepts the equilibrium hypotheses, he plays the equilibrium
  - The goal of the equilibrium hypothesis is that we do not stray from equilibrium because AWESOME plays the best response
- Precompute equilibrium
- Restart means forgetting everything
- When this strategy is rejected, AWESOME picks a random action
- If another action appears to be **significant** better, AWESOME will pick that one
  - Significant difference is important to prohibit AWESOME from jumping back and forth

First Approach



- Apply same test of hypothesis every epoch:
  - Same number of rounds per epoch
  - If observed distribution of actions differs form hypothesized distribution, reject

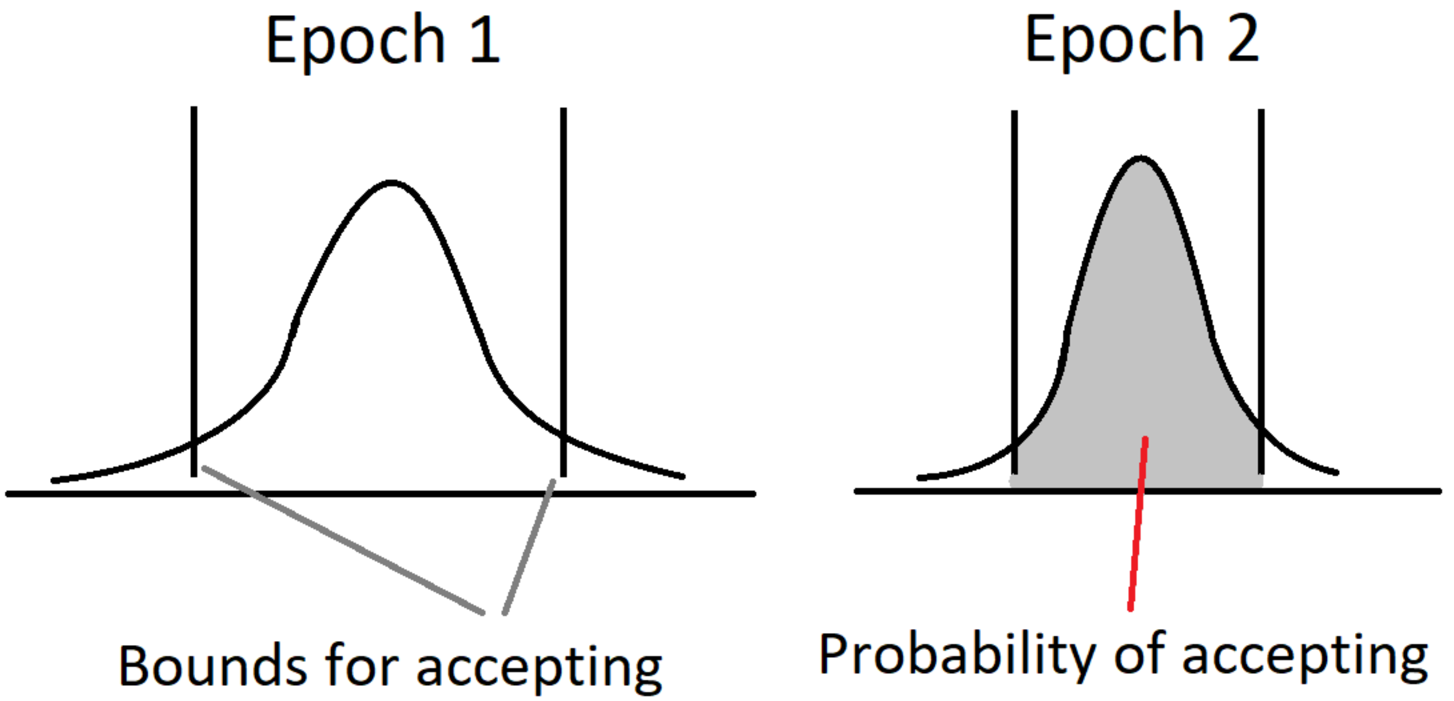
$$|p_h - p_{\pi^*}| < \epsilon_e$$

$$|p_h - p_{h^{prev}}| < \epsilon_s$$

- Problems:
  - Each epoch there is fixed probability of rejecting
  - Distinguish a distribution within a epsilon from hypothesized

Solution

- Simple: increase epoch length N, decrease threshold
  - Observation should get closer to hypothesized one



The Algorithm

```
AWESOME()
for each $p$
  $\pi^*_p$ := ComputeEquilibriumStrategy($p$)
repeat {// beginning of each restart
  for each player $p$ {
    InitializeToEmpty($h^{prev}_p$)
    InitializeToEmpty($h^{curr}_p$) }
  $APPE := true$
  $APS := true$
  $\delta := false$
  $t := 0$
  $\phi := \pi^*_{Me}$
  while $APS$ { // beginning of each epoch
    repeat $N^t$ times {
      Play($\phi$)
      for each player $p$
        Update($h^{curr}_p$) }
    if $APPE = false$ {
     if $\delta = false$
      for each player $p$
       if (Distance($h^{curr}_p$, $h^{prev}_p$) > $\epsilon^t_s$)
        $APS := false$
     $\delta := false$
     $a := \arg\max V(a, h^{curr}_{-Me})$
     if $V(a, h^{curr}_{-Me}) > V(\phi, h^{curr}_{-Me}) + n|A|\epsilon^{t+1}_s\mu$
       $\phi := a$ }
    if $APPE = true$
     for each player $p$
      if (Distance($h^{curr}_p$, $\pi^*_p$) > $\epsilon^t_e$) {
        $APPE := false$
        $\phi :=$ RandomAction()
        $\delta := true$ }
    for each player $p$ {
      $h^{prev}_p := h^{curr}_p$
      InitializeToEmpty($h^{curr}_p$) }
    $t := t + 1$ } }
```

[2]

AWESOME()

for each p
$\quad \pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat { // beginning of each restart
$\quad$ **for each** player p {
$\quad\quad$ InitializeToEmpty(h_p^{prev})
$\quad\quad$ InitializeToEmpty(h_p^{curr}) }
$\quad APPE := true$
$\quad APS := true$
$\quad \delta := false$
$\quad t := 0$
$\quad \phi := \pi_{Me}^*$
$\quad$ **while** APS { // beginning of each epoch
$\quad\quad$ **repeat** N^t **times** {
$\quad\quad\quad$ Play(ϕ)
$\quad\quad\quad$ **for each** player p
$\quad\quad\quad\quad$ Update(h_p^{curr}) }
$\quad\quad$ **if** $APPE = false$ {
$\quad\quad\quad$ **if** $\delta = false$
$\quad\quad\quad\quad$ **for each** player p
$\quad\quad\quad\quad\quad$ **if** (Distance$(h_p^{curr}, h_p^{prev}) > \epsilon_s^t$)
$\quad\quad\quad\quad\quad\quad APS := false$
$\quad\quad\quad \delta := false$
$\quad\quad\quad a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
$\quad\quad\quad$ **if** $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\quad\quad\quad\quad \phi := a$ }
$\quad\quad$ **if** $APPE = true$
$\quad\quad\quad$ **for each** player p
$\quad\quad\quad\quad$ **if** (Distance$(h_p^{curr}, \pi_p^*) > \epsilon_e^t$) {
$\quad\quad\quad\quad\quad APPE := false$
$\quad\quad\quad\quad\quad \phi :=$ RandomAction()
$\quad\quad\quad\quad\quad \delta := true$ }
$\quad\quad$ **for each** player p {
$\quad\quad\quad h_p^{prev} := h_p^{curr}$
$\quad\quad\quad$ InitializeToEmpty(h_p^{curr}) }
$\quad\quad t := t + 1$ } }

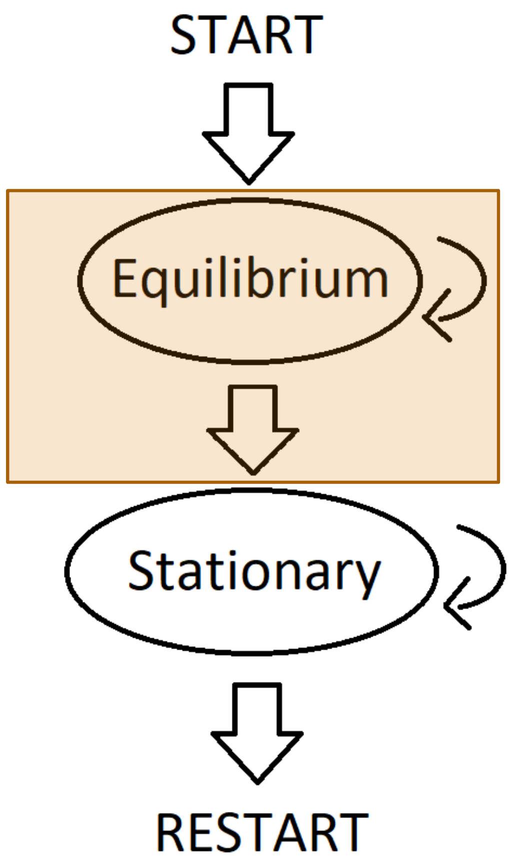
AWESOME()

for each p
$\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
for each player p {
InitializeToEmpty(h_p^{prev})
InitializeToEmpty(h_p^{curr}) }
$APPE := true$
$APS := true$
$\delta := false$
$t := 0$
$\phi := \pi_{Me}^*$
while APS { // beginning of each epoch
repeat N^t **times** {
Play(ϕ)
for each player p
Update(h_p^{curr}) }
if $APPE = false$ {
if $\delta = false$
for each player p
if (Distance(h_p^{curr}, h_p^{prev}) $> \epsilon_s^t$)
$APS := false$
$\delta := false$
$a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
if $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\phi := a$ }
if $APPE = true$
for each player p
if (Distance(h_p^{curr}, π_p^*) $> \epsilon_e^t$) {
$APPE := false$
$\phi :=$ RandomAction()
$\delta := true$ }
for each player p {
$h_p^{prev} := h_p^{curr}$
InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }

AWESOME()

for each p
$\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
for each player p {
InitializeToEmpty(h_p^{prev})
InitializeToEmpty(h_p^{curr}) }
$APPE := true$
$APS := true$
$\delta := false$
$t := 0$
$\phi := \pi_{Me}^*$
while APS { // beginning of each epoch
repeat N^t **times** {
Play(ϕ)
for each player p
Update(h_p^{curr}) }
if $APPE = false$ {
if $\delta = false$
for each player p
if (Distance$(h_p^{curr}, h_p^{prev}) > \epsilon_s^t$)
$APS := false$
$\delta := false$
$a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
if $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\phi := a$ }
if $APPE = true$
for each player p
if (Distance$(h_p^{curr}, \pi_p^*) > \epsilon_e^t$) {
$APPE := false$
$\phi :=$ RandomAction()
$\delta := true$ }
for each player p {
$h_p^{prev} := h_p^{curr}$
InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }

for each player p
  if (Distance(h_p^{curr}, h_p^{prev}) $> \epsilon_s^t$)
    $APS := \mathit{false}$
$\delta := \mathit{false}$
$a := \arg\max \mathrm{V}(a,\, h_{-Me}^{curr})$
if $\mathrm{V}(a,\, h_{-Me}^{curr}) > \mathrm{V}(\phi,\, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
  $\phi := a$ }
if $APPE = \mathit{true}$
 for each player p
  if (Distance(h_p^{curr}, π_p^*) $> \epsilon_e^t$) {
    $APPE := \mathit{false}$
    $\phi :=$ RandomAction()
    $\delta := \mathit{true}$ }
for each player p {
  $h_p^{prev} := h_p^{curr}$
  InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }



AWESOME()

for each p
$\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
for each player p {
InitializeToEmpty(h_p^{prev})
InitializeToEmpty(h_p^{curr}) }
$APPE := true$
$APS := true$
$\delta := false$
$t := 0$
$\phi := \pi_{Me}^*$
while APS { // beginning of each epoch
repeat N^t **times** {
Play(ϕ)
for each player p
Update(h_p^{curr}) }
if $APPE = false$ {
if $\delta = false$
for each player p
if (Distance(h_p^{curr}, h_p^{prev}) $> \epsilon_s^t$)
$APS := false$
$\delta := false$
$a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
if $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\phi := a$ }
if $APPE = true$
for each player p
if (Distance(h_p^{curr}, π_p^*) $> \epsilon_e^t$) {
$APPE := false$
$\phi :=$ RandomAction()
$\delta := true$ }
for each player p {
$h_p^{prev} := h_p^{curr}$
InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }

AWESOME()

for each p
$\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
for each player p {
InitializeToEmpty(h_p^{prev})
InitializeToEmpty(h_p^{curr}) }
$APPE := true$
$APS := true$
$\delta := false$
$t := 0$
$\phi := \pi_{Me}^*$
while APS { // beginning of each epoch
repeat N^t **times** {
Play(ϕ)
for each player p
Update(h_p^{curr}) }
if $APPE = false$ {
if $\delta = false$
for each player p
if (Distance$(h_p^{curr}, h_p^{prev}) > \epsilon_s^t$)
$APS := false$
$\delta := false$
$a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
if $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\phi := a$ }
if $APPE = true$
for each player p
if (Distance$(h_p^{curr}, \pi_p^*) > \epsilon_e^t$) {
$APPE := false$
$\phi :=$ RandomAction()
$\delta := true$ }
for each player p {
$h_p^{prev} := h_p^{curr}$
InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }

AWESOME()

for each p
  $\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
  for each player p {
    InitializeToEmpty(h_p^{prev})
    InitializeToEmpty(h_p^{curr}) }
  $APPE := true$
  $APS := true$
  $\delta := false$
  $t := 0$
  $\phi := \pi_{Me}^*$
  while APS { // beginning of each epoch
    repeat N^t **times** {
      Play(ϕ)
      for each player p
        Update(h_p^{curr}) }
    if $APPE = false$ {
     if $\delta = false$
      for each player p
       if $(\text{Distance}(h_p^{curr}, h_p^{prev}) > \epsilon_s^t)$
        $APS := false$
     $\delta := false$
     $a := \arg\max \text{V}(a, h_{-Me}^{curr})$
     if $\text{V}(a, h_{-Me}^{curr}) > \text{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
      $\phi := a$ }
    if $APPE = true$
     for each player p
      if $(\text{Distance}(h_p^{curr}, \pi_p^*) > \epsilon_e^t)$ {
       $APPE := false$
       $\phi :=$ RandomAction()
       $\delta := true$ }
    for each player p {
     $h_p^{prev} := h_p^{curr}$
     InitializeToEmpty(h_p^{curr}) }
    $t := t + 1$ } }

		100%	
	1, 2	2, 1	3, 0
	2, 1	1, 2	4, 0
100%	0, 3	0, 4	2, 2

		100%	
100%	1, 2	2, 1	3, 0
	2, 1	1, 2	4, 0
	0, 3	0, 4	2, 2

AWESOME()

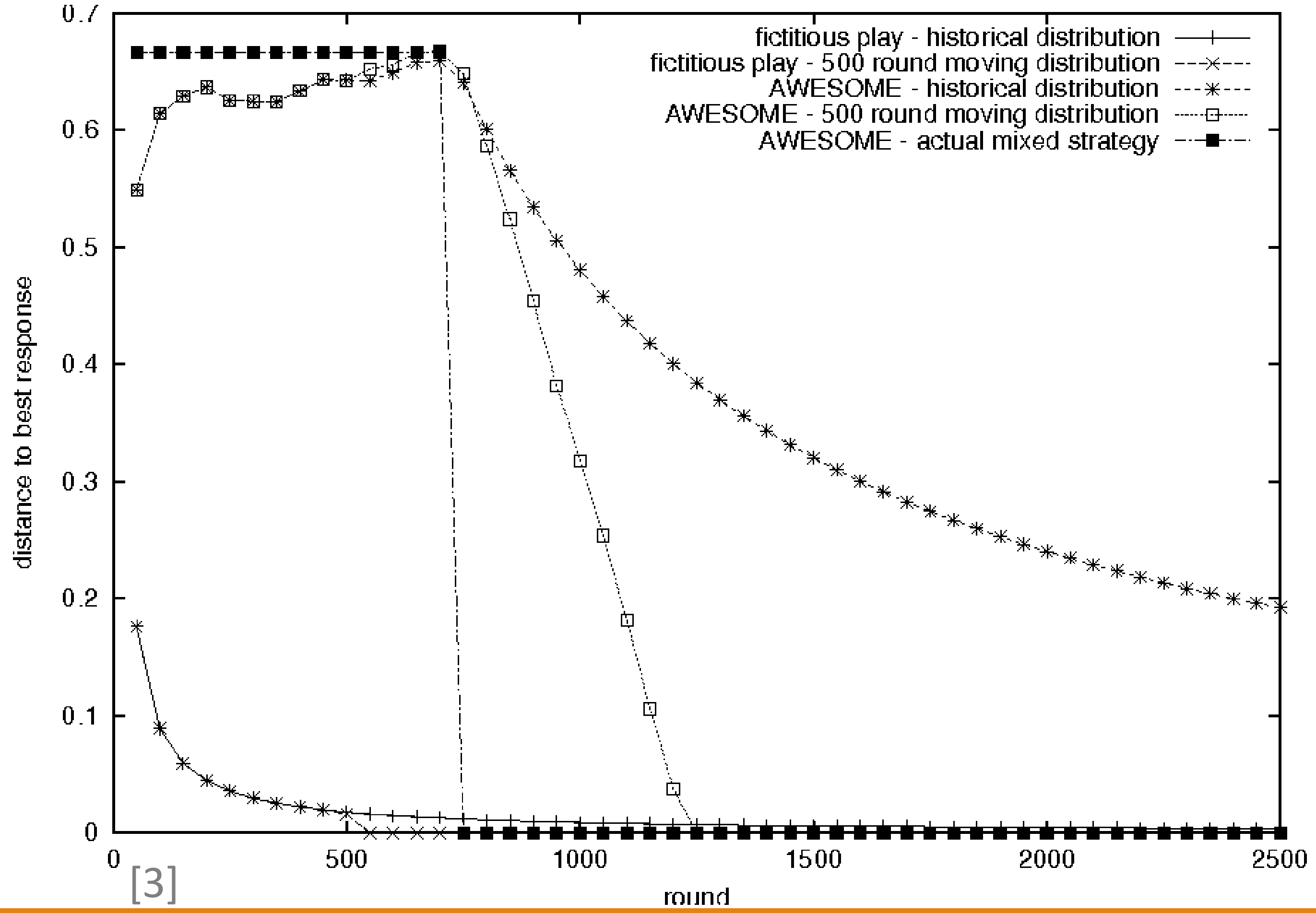
for each p
$\pi_p^* :=$ ComputeEquilibriumStrategy(p)
repeat {// beginning of each restart
for each player p {
InitializeToEmpty(h_p^{prev})
InitializeToEmpty(h_p^{curr}) }
$APPE := true$
$APS := true$
$\delta := false$
$t := 0$
$\phi := \pi_{Me}^*$
while APS { // beginning of each epoch
repeat N^t **times** {
Play(ϕ)
for each player p
Update(h_p^{curr}) }
if $APPE = false$ {
if $\delta = false$
for each player p
if (Distance(h_p^{curr}, h_p^{prev}) $> \epsilon_s^t$)
$APS := false$
$\delta := false$
$a := \arg\max \mathrm{V}(a, h_{-Me}^{curr})$
if $\mathrm{V}(a, h_{-Me}^{curr}) > \mathrm{V}(\phi, h_{-Me}^{curr}) + n|A|\epsilon_s^{t+1}\mu$
$\phi := a$ }
if $APPE = true$
for each player p
if (Distance(h_p^{curr}, π_p^*) $> \epsilon_e^t$) {
$APPE := false$
$\phi :=$ RandomAction()
$\delta := true$ }
for each player p {
$h_p^{prev} := h_p^{curr}$
InitializeToEmpty(h_p^{curr}) }
$t := t + 1$ } }

Comparison 1 – Stationary Opponent

- Fictitious play
  - Very simple learning algorithm
  - Plays best response for history

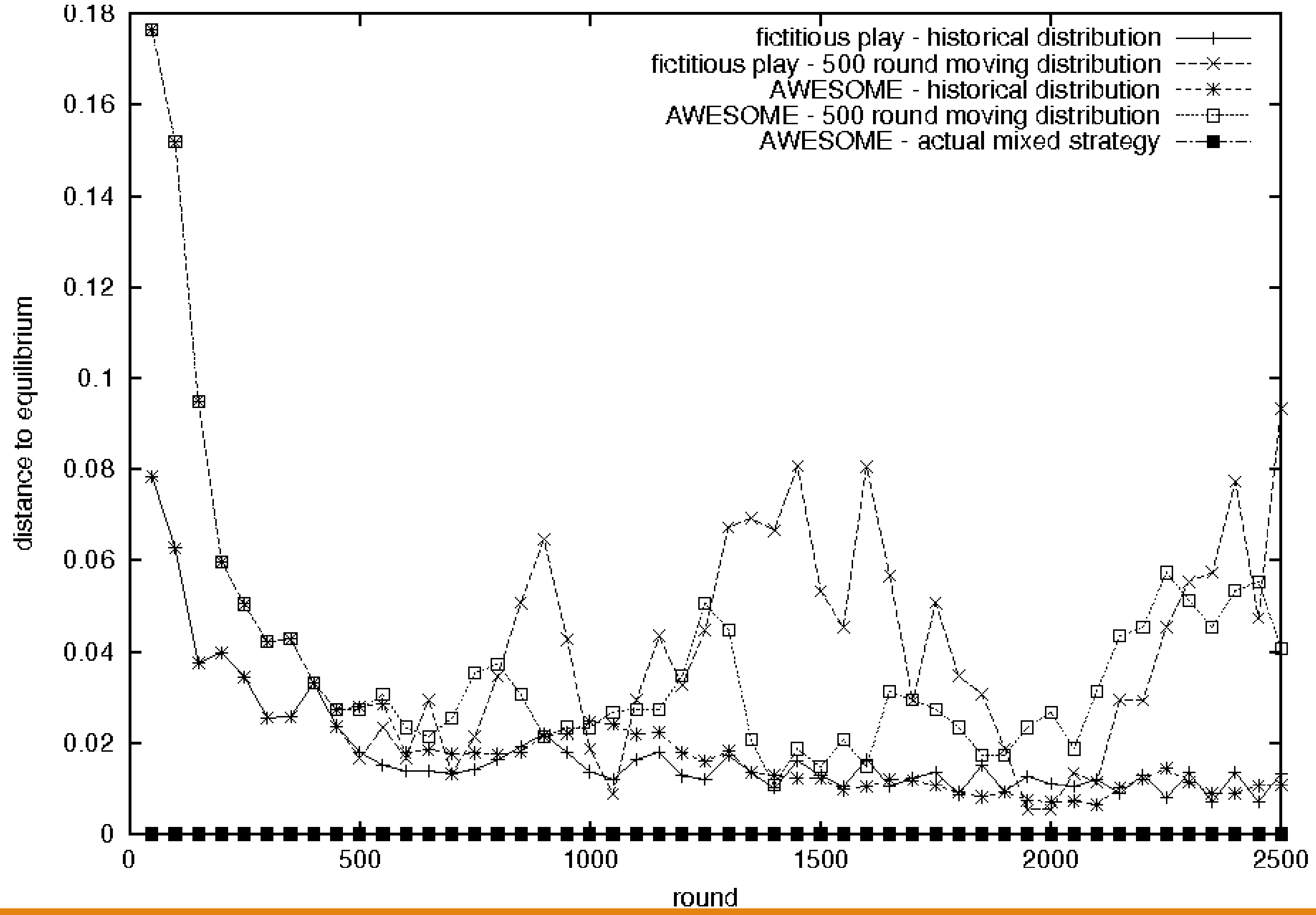
- Rock-Paper-Scissors
  - Mixed Strategy (0.4, 0.6, 0)

	R	P	S
R	.5, .5	0, 1	1, 0
P	1, 0	.5, .5	0, 1
S	0, 1	1, 0	.5, .5



[3]

Comparison 1 – Self-Play

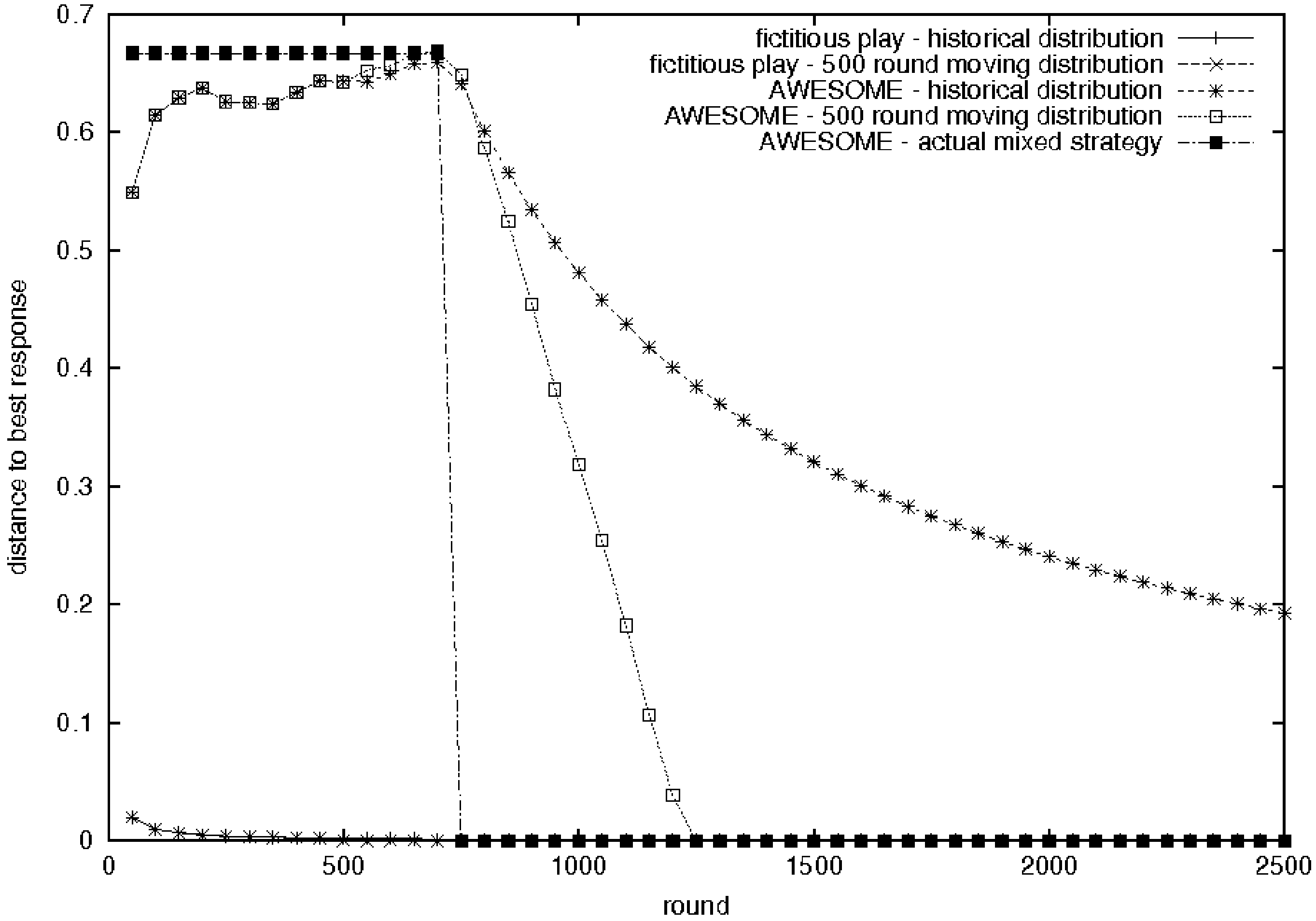


[3]

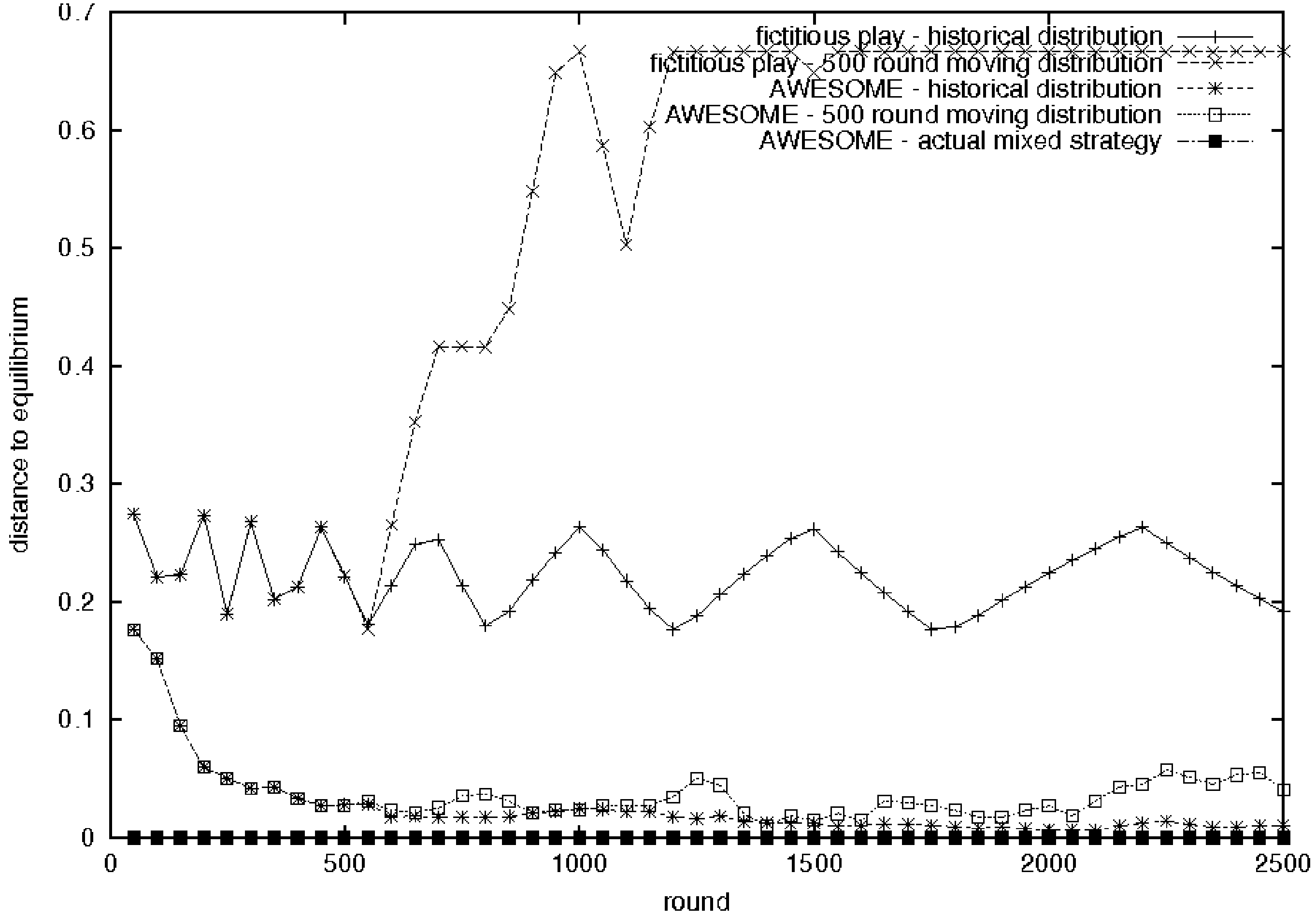
Comparison 2 – Stationary Opponent

- Shapley‘s game
  - Mixed strategy (0.4, 0.6, 0)

0, 1	0, 0	1, 0
0, 0	1, 0	0, 1
1, 0	0, 1	0, 0



Comparison 2 - Self-Play



Summary

- AWESOME is algorithm for learning in repeated games:
  - Best response against stationary opponents
  - Nash equilibrium in self-play

- Try to adapt when everyone is stationary, otherwise play equilibrium

- Achieves this by testing hypotheses in each epoch

References

[1] 10.06.19
https://www.mathematik.de/spudema/spudema_beitraege/beitraege/kuhlenschmidt/nash.htm
[2] Conitzer, V.; Sandholm T.: AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents, 2003
[3] Conitzer, V.; Sandholm T.: AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents, 2007
[4] 10.06.19 http://www.claxonmarketing.com/2014/02/16/is-awesome-awesome/
[5] 12.06.19 https://www.pinterest.de/pin/450852612673884638/?lp=true
[6] 12.06.19 https://lifestyle.howstuffworks.com/crafts/seasonal/baseball-activities1.htm

Thank you for your attention!