

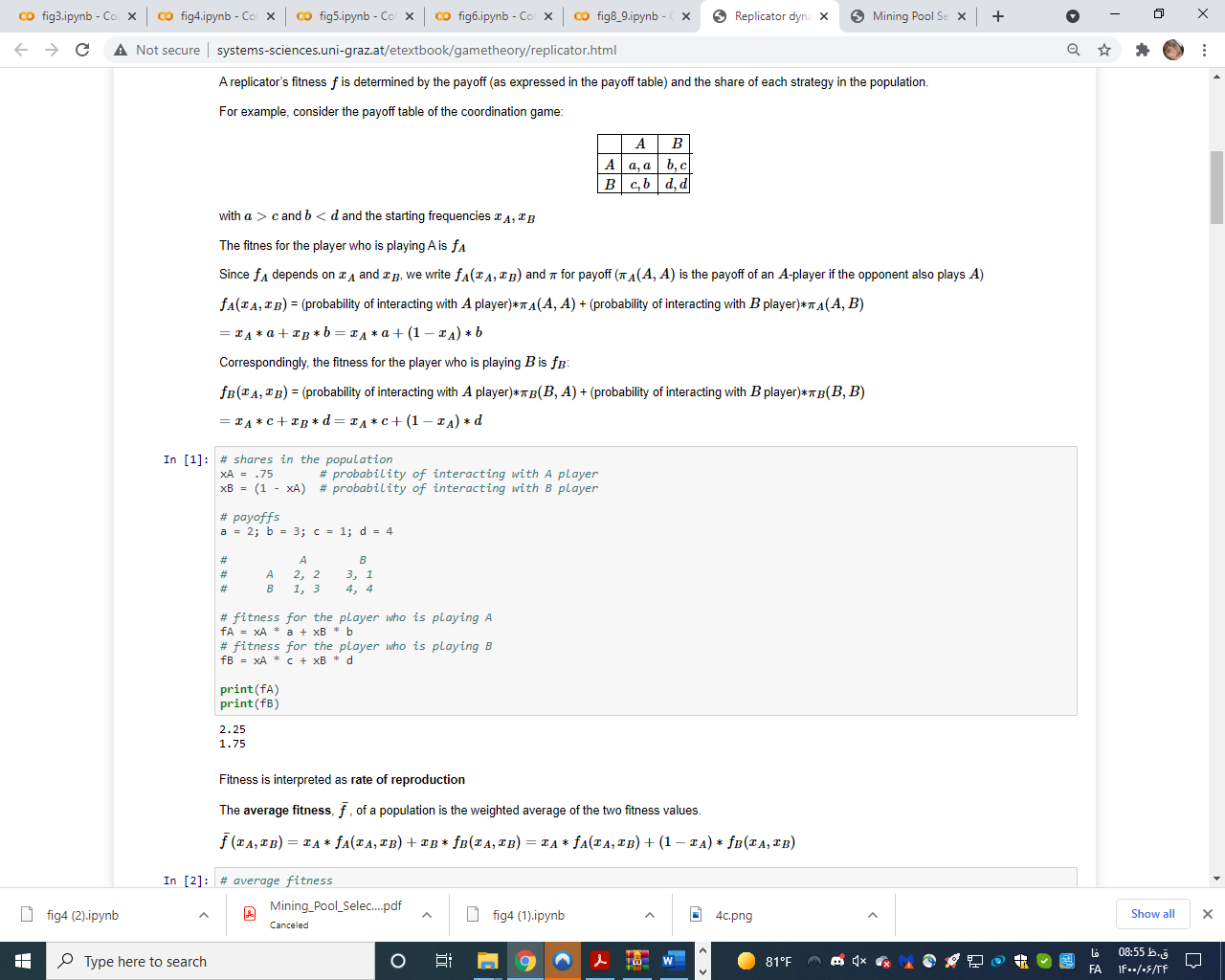
**گزارش پیاده سازی مقاله برای درس نظریه بازی ها**

**یاسمن صفاری**

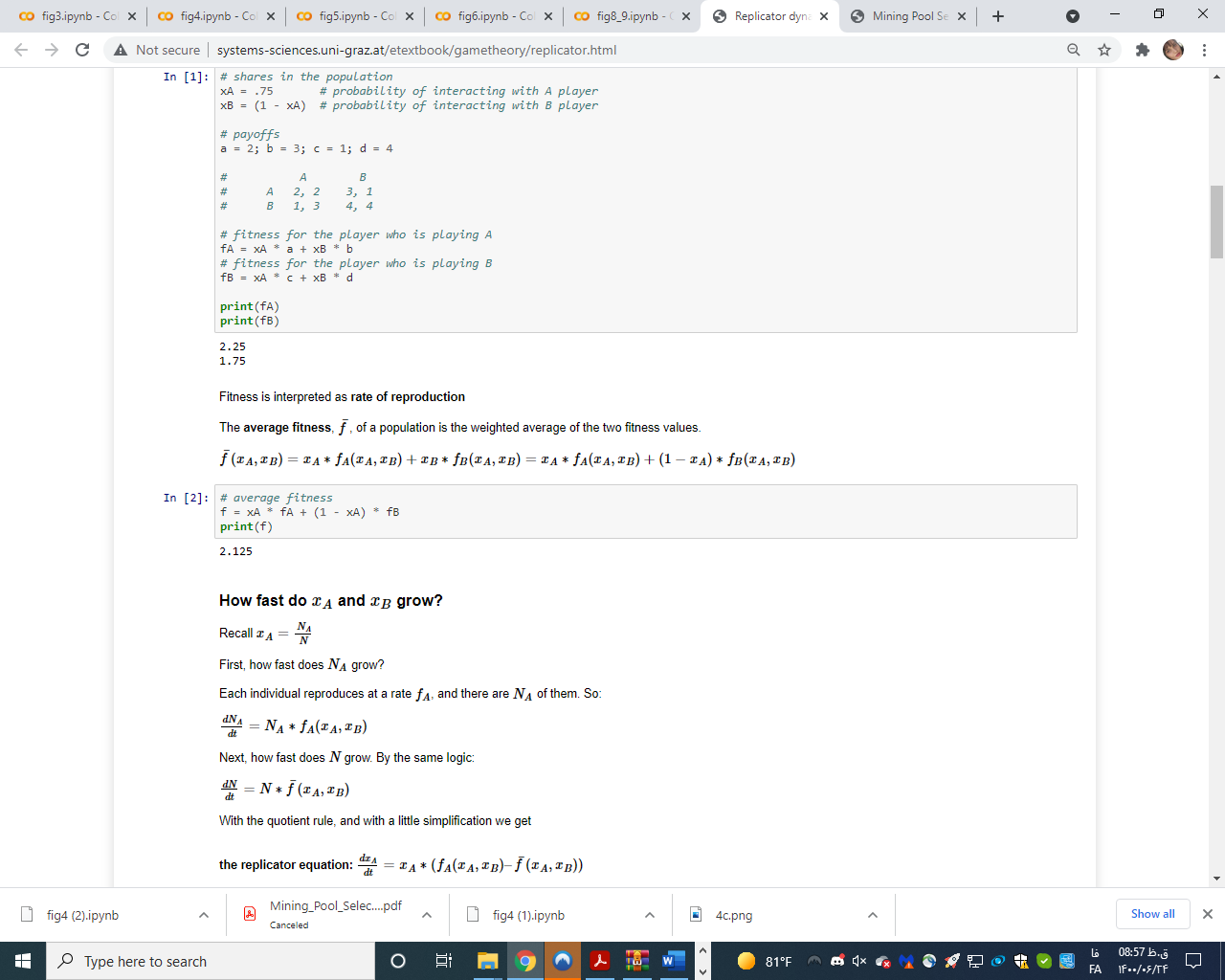
**هدف**: یافتن ESS برای جمعیت pool ها در وضعیت وجود حمله BWA

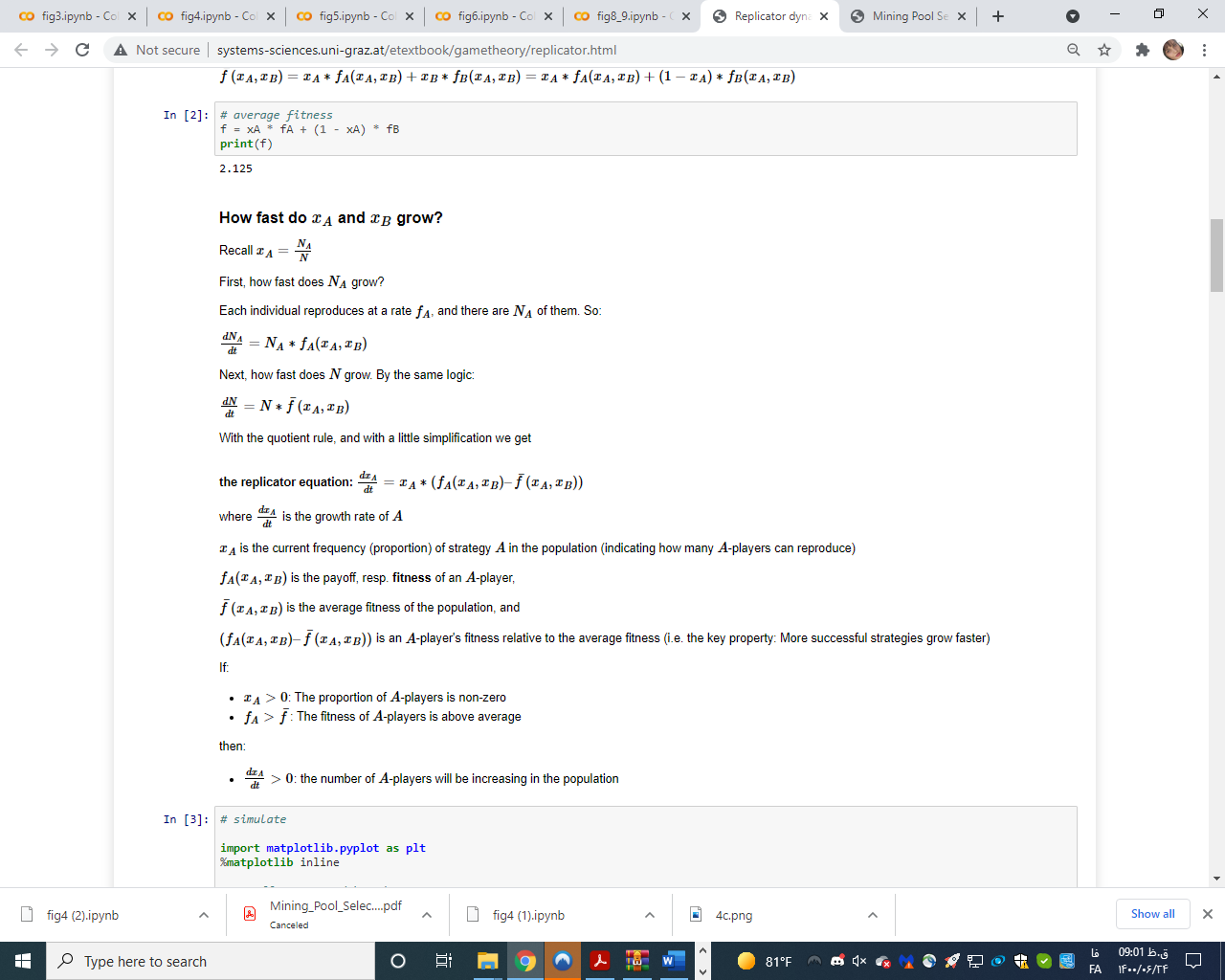
**چکیده**: در شبکه بلاکچین این مقاله، ماینرها برای دریافت پاداش بهتر بابت مشارکتشان در شرایطی که استخرها شامل حملات هستند، میتوانند از استخری به استخر دیگر مهاجرت کنند. در این مقاله برای بررسی شرایط پویای استخرها از نظر جمعیت ماینرها از نظریه بازی های تکاملی استفاده شد. برای این منظور پس از تعریف مدل پاداش برای استخر های حمله کننده یا مورد حمله قرار گرفته، به کمک رپلیکاتور و بررسی نرخ رشد جمعیت استخرها، ESS های مسیله بدست آورده شدند. همینطور معیار سایز بلاک های ماینینگ به عنوان یکی از معیارهای موثر در تعیین ess ها مورد بررسی قرار گرفت تا یک یادگیری پاسخ بهتر ساخته شود.

**مقدمه**: بخشی از دانش لازم برای فهم مقاله مربوط به مفاهیم گیم تیوری مطرح شده در کلاس است که بازگو نمیکنم. اما از آنجا که این مقاله از روش نظریه تکاملی استفاده کرده است مختصر به این مبحث میپردازیم. نظریه تکاملی بازی ها از مبحث تکامل در طبیعت نشات میگیرد و معمولا با جمعیت ها و بررسی تغییرات جمعیت در طول زمان سر و کار دارد. توضیح را با یک مثال کلی ادامه میدهم.

فرض کنید n نفر به طور کل وجود دارد. استراتژی های بازی در این مسایل تقسیم کننده افراد خواهد بود. به طور مثال دو استراتژی حمله کننده و دفاع داشته باشیم. تعداد استراتژی ها محدود به دو نیست و ما برای سادگی دو را انتخاب میکنیم. نسبت جمعیتی که هر استراتژی را بازی میکند ابتدا مقدار دهی اولیه میکنیم. در این مسایل با نسبت جمعیت و نرخ رشد سروکار داریم. در این مسیله یک استراتژی پروفایل، نسبت جمعیت متعلق به هر استراتژی است که مجموع آن یک است. حال باید بر اساس روابط پاداش و یا جدول پاداشها، مقدار فیتنس یا مطلوبیت یا همان وزن هر استراتژی که در رشد آنها در نسل اینده موثر است را محاسبه کنیم. 

فیتنس تمام استراتژی ها محاسبه میشوند. حال فیتنس میانگین همه استراتژی ها هم محاسبه میکنیم.



برای تولید نسل های بعدی، مهمترین چیز دانستن نرخ یا سرعت رشد نسبت جمعیت استراتژی ها است. برای این منظور معادله رپلیکاتور یا همان نرخ رشد رامحاسبه میکنیم. 

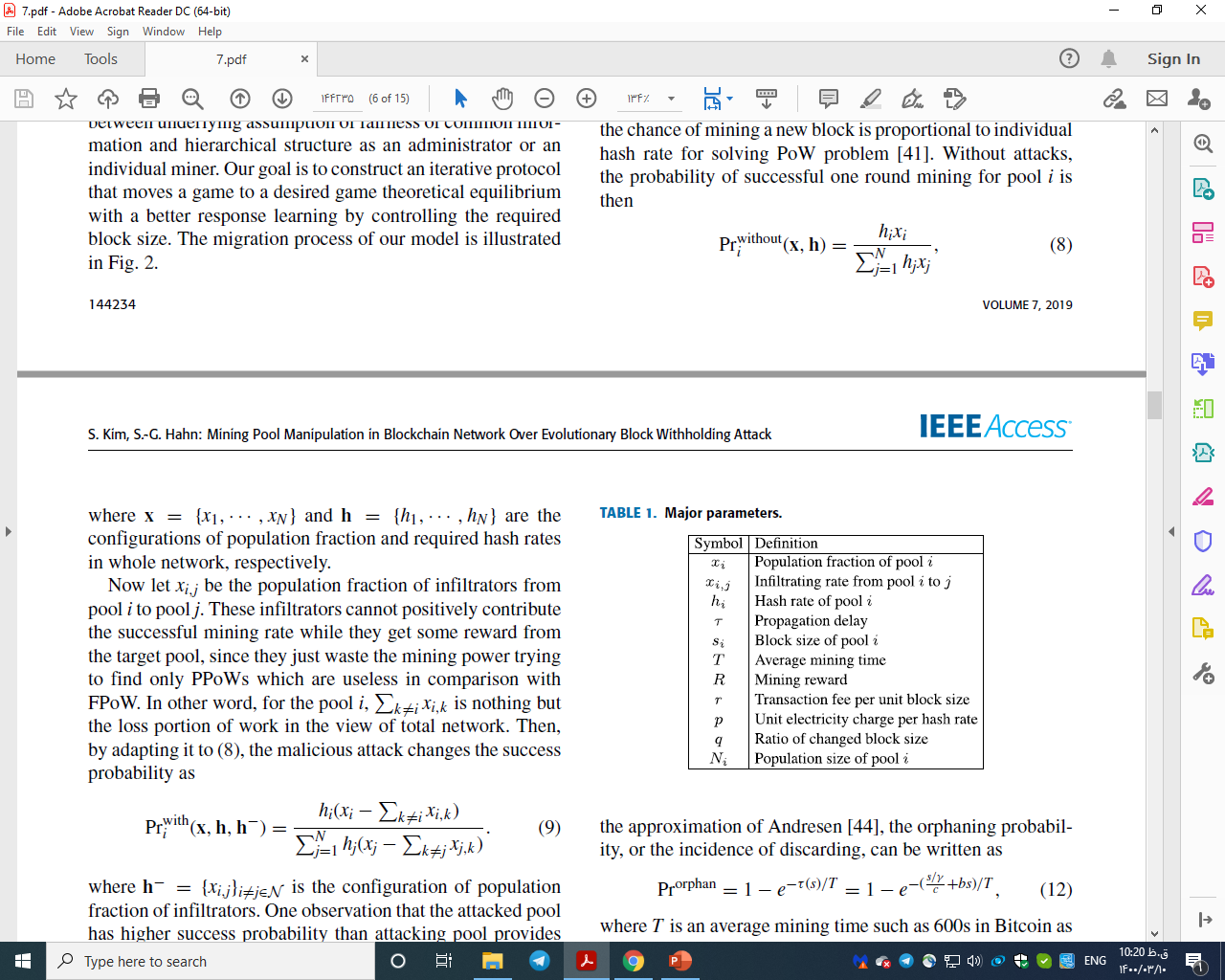
حال نسبت جدید جمعیت هر استراتژی را محاسبه میکنیم. برای اینکار نسبت جمعیت فعلی را با مقدار به دست امده از معادله رپلیکاتور باید جمع کنیم. نسبت های جدید نیز مجموعشان یک خواهد بود. این روند تکامل را در طول زمان بارها تکرار میکنیم تا نقاط تعادل را بررسی کنیم.

برای یافتن فیک پوینتها یا نش ها باید رپلیکاتور همه استراتژی ها را برابر صفر قرار داده و نسبت جمعیت ها را محاسبه کرد. برای بررسی پایداری نقاط ب دست امده باید مطمین شوید که اگر یک اپسیلون نسبت های نش بدست امده تغییر کنند، در طول زمان مجدد نسبت ها به نش برگردند در غیر اینصورت پایدار و ess نیست.

در این مقاله استراتژی ها حمله کننده و مورد حمله است که عملا ویژگی استخرهایی است که ماینرها مهاجرت میکنند. معادلات پاداش مقاله بر اساس معیارهای بلاکچینها مانند نرخ هش و سایز بلاک و نرخ حمله و... بدست آمده است. ماینرها قصد دارند تا برای ماکزیمم کردن پاداش خودشان بر اساس معادله پاداش مهاجرت کنند. پس در واقع با مساله حل یک مساله معادله دیفرانسیل نسبت به جمعیت سروکار داریم.

**پیاده سازی**

برای پیاده سازی با زبان پایتون و در کولب انجام شد. برای پیاده سازی از هیچ فریم ورک یا ابزار خاصی استفاده نشد و کدها خام پیاده شدند. در جدول زیر نمادها اورده شده اند.



برای راحتتر بودن به کد زیر توجه کنید . بخش های عمده مراحل پیاده سازی از تعریف توابع تا بررسی نش ها مشخص است.

# simulate

import matplotlib.pyplot as plt

import math

%matplotlib inline

# مقداردهی اولیه پارامترها که مقادیر و معنای نماد ها در مقاله مشخص است

h1 = 2; h2 = 1; x12 = 0.5; x21=0.3 ;ta=20 ;r=6 ;T=600 ;p=0.1

#میخواهیم بعدا نمودار رشد جمعیت ها را رسم کنیم. لذا در هر جنریشن، نرخ جمعیت جدید را در لیستی ثبت میکنیم.

x1 = [0.7]

x2 = [1 - x1[0]]

#نرخ گام زمانی مربوط به دیفرانسیل که به طور تجربی تعیین کردم

dt = 0.001

# معادلات فینس هر دو استراتژی را مینویسیم. معادلات دقیقا در مقاله بیان شده اند.

#همانطور که میبینید فیتنس اولیه در یک لیست معرفی خواهد شد چرا که بعد در طول لوپ قراره  فیتنسهای هر دوره ثبت شوند

F1 = [(((h1\*(x1[0]-x12))/((h1\*(x1[0]-x12))+(h2\*(x2[0]-x21))))\*(r\*(math.exp(-(ta/T))))-(p\*h1\*(x1[0]-x12))) \* dt]

F2 = [(((h2\*(x2[0]-x21))/((h1\*(x1[0]-x12))+(h2\*(x2[0]-x21))))\*(r\*(math.exp(-(ta/T))))-(p\*h2\*(x2[0]-x21))) \* dt]

#محاسبه فیتنس میانگین

F = [(x1[0] \* F1[0] + x2[0] \* F2[0]) \* dt]

#یک تعداد دور جنریشن را تعیین میکنیم و هر بار رپلیکاتورها محاسبه و نرخ جمعیت های جدید به دست امده و در لیست مربوطه ذخیره میشوند.

for t in range(1000):

    # محاسبه فیتنس این زمان

    f1 = ((((h1\*(x1[t]-x12))/((h1\*(x1[t]-x12))+(h2\*(x2[t]-x21))))\*(r)\*(math.exp(-(ta/T))))-(p\*h1\*(x1[t]-x12)))

    f2 = ((((h2\*(x2[t]-x21))/((h1\*(x1[t]-x12))+(h2\*(x2[t]-x21))))\*(r)\*(math.exp(-(ta/T))))-(p\*h2\*(x2[t]-x21)))

    #معادله فیتنس میانگین

    ff = x1[t] \* f1 + x2[t] \* f2

    #افزودن فیتنس ها و میانگین به لیستهایی ک قبلا ساختیم

    F1.append(f1\*dt)

    F2.append(f2\*dt)

    F.append(ff\*dt)

    # محاسبه نرخ جمعیت های جدید که حاصل جمع نرخ فعلی با رپلیکاتور است. و اضافه کردن نتیجه به لیست مربوطه

    x1.append(x1[t] + ((x1[t]\*(1-x1[t])\*(f1-f2)) \* dt))

    x2.append(x2[t] + ((x2[t]\*(1-x2[t])\*(f2-f1)) \* dt))

    #x1.append(x1[t] + ((x1[t]\*(f1-ff)) \* dt))

    #x2.append(x2[t] + ((x2[t]\*(f2-ff)) \* dt))

# رسم نمودار لیست هایی ک تهیه کردیم ( نسبت جمعیت هر استراتژی، فیتنس از استراتژی، فیتنس میانگین)

plt.plot(x1, 'r', label = 'share of poo1')

plt.plot(x2, 'b', label = 'share of poo2')

#plt.plot(F1, 'r--', label ='fitness of cooperators')

#plt.plot(F2, 'b--', label ='fitness of defectors')

#plt.plot(F, 'g--', label ='mean population fitness')

plt.legend(loc = 'best')

plt.ylim(-0.1, 1.1)

plt.grid()

نتیجه اجرای کد بالا را با تنظیمات مختلفی که مقاله گفته بود به دست اوردیم که بعدا گزارش میشود. فعلا هدف بررسی نحوه پیاده است. لذا حال نوبت اینست که نقاط نش محاسبه شوند.

# use sympy to calculate steady states

from sympy import \*

import math

# paras

h1 = 2; h2 = 1; x12 = 0.5; x21=0.3 ;ta=20 ;r=6 ;T=600 ;p=0.1

dt = 0.001

#نوشتن معادلات فیتنس و میانگین فیتنس بر حسب متغیر نسبت جمعیت ها. متوجهید ک اینجا نباید مقدار دهی اولیه به نسبت جمعیتها کرد.

x1, x2 = symbols('x1, x2')

f1 = (((h1\*(x1-x12))/((h1\*(x1-x12))+(h2\*(x2-x21))))\*(r\*(math.exp(-(ta/T))))-(p\*h1\*(x1-x12)))

f2 = (((h2\*(x2-x21))/((h1\*(x1-x12))+(h2\*(x2-x21))))\*(r\*(math.exp(-(ta/T))))-(p\*h2\*(x2-x21)))

f = (x1 \* f1 + x2 \* f2)

#نوشتن معادله رپلیکاتور هر استراتژی

d1 = x1 \* (f1-f)

d2 = x2 \* (f2-f)

#صفر قرار دادن رپلیکاتورها

# use sympy's way of setting equations to zero

Equal1 = Eq(d1, 0)

Equal2 = Eq(d2, 0)

print(Equal1)

#حل معادلات رپلیکاتور نسبت به متغیرهای نسبت جمعیتها

# compute fixed points

equilibria = solve([Equal1, Equal2], [x1, x2])

print(equilibria)

#حال تمام جوابهای ممکن را دارید و میتوانید بررسی کنید

برای بررسی پایداری نقاط نش: # simulate with steady state

import matplotlib.pyplot as plt

%matplotlib inline

# paras

h1 = 2; h2 = 1; x12 = 0.5; x21=0.3 ;ta=20 ;r=6 ;T=600 ;p=0.1

# یک مقدار اندکی در نقطه نش مورد نظرتون دستکاری کنید.

x1 = [1 - 0.00001]

x2 = [1 - x1[0]]

dt = 0.001

# به تعداد دور مناسب، با کمک معادلات فیتنس و رپلیکاتور نسل سازی کرده و نمودار نسبت های جمعیت جدید را رسم کنید.

for t in range(30000):

    # fitnesses

    f1 = ((((h1\*(x1[t]-x12))/((h1\*(x1[t]-x12))+(h2\*(x2[t]-x21))))\*(r)\*(math.exp(-(ta/T))))-(p\*h1\*(x1[t]-x12)))

    f2 = ((((h2\*(x2[t]-x21))/((h1\*(x1[t]-x12))+(h2\*(x2[t]-x21))))\*(r)\*(math.exp(-(ta/T))))-(p\*h2\*(x2[t]-x21)))

    ff = x1[t] \* f1 + x2[t] \* f2

    # differential equations for shares

    x1.append(x1[t] + ((x1[t]\*(1-x1[t])\*(f1-f2)) \* dt))

    x2.append(x2[t] + ((x2[t]\*(1-x2[t])\*(f2-f1)) \* dt))

plt.plot(x1, 'r', label = 'share of pool1')

plt.plot(x2, 'b', label = 'share of pool2')

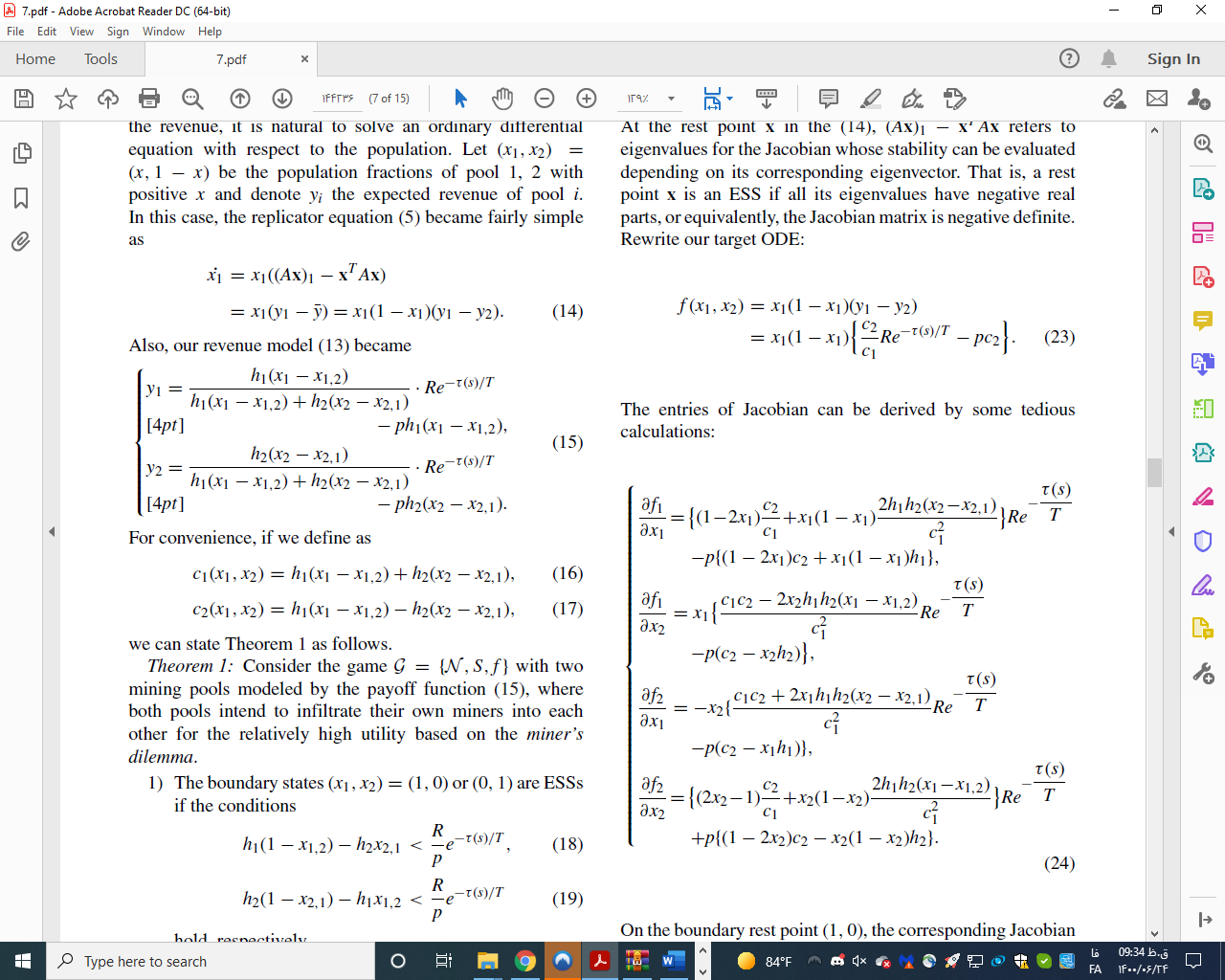
plt.legend(loc = 'best')

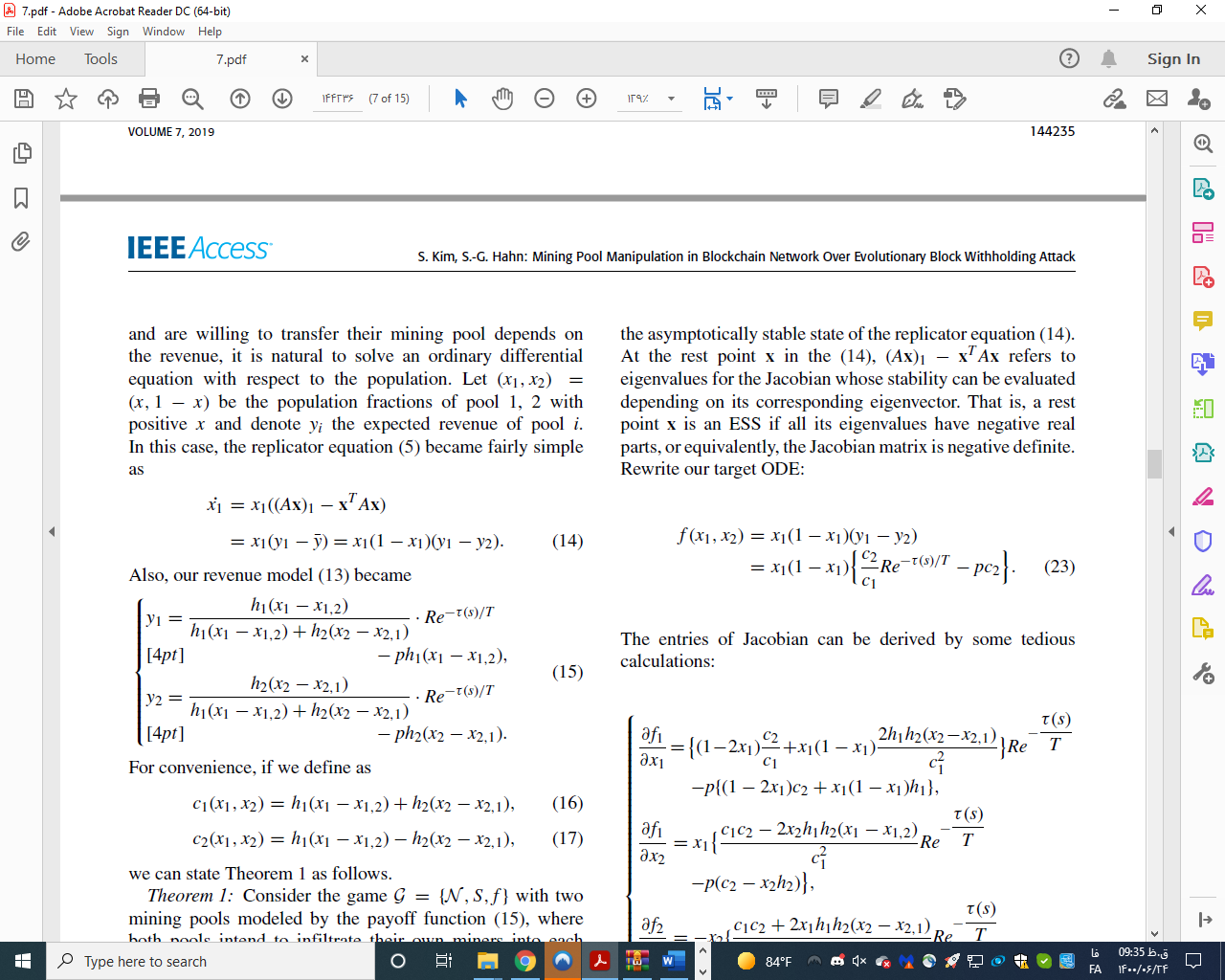
plt.ylim(-0.1, 1.1)

plt.grid()

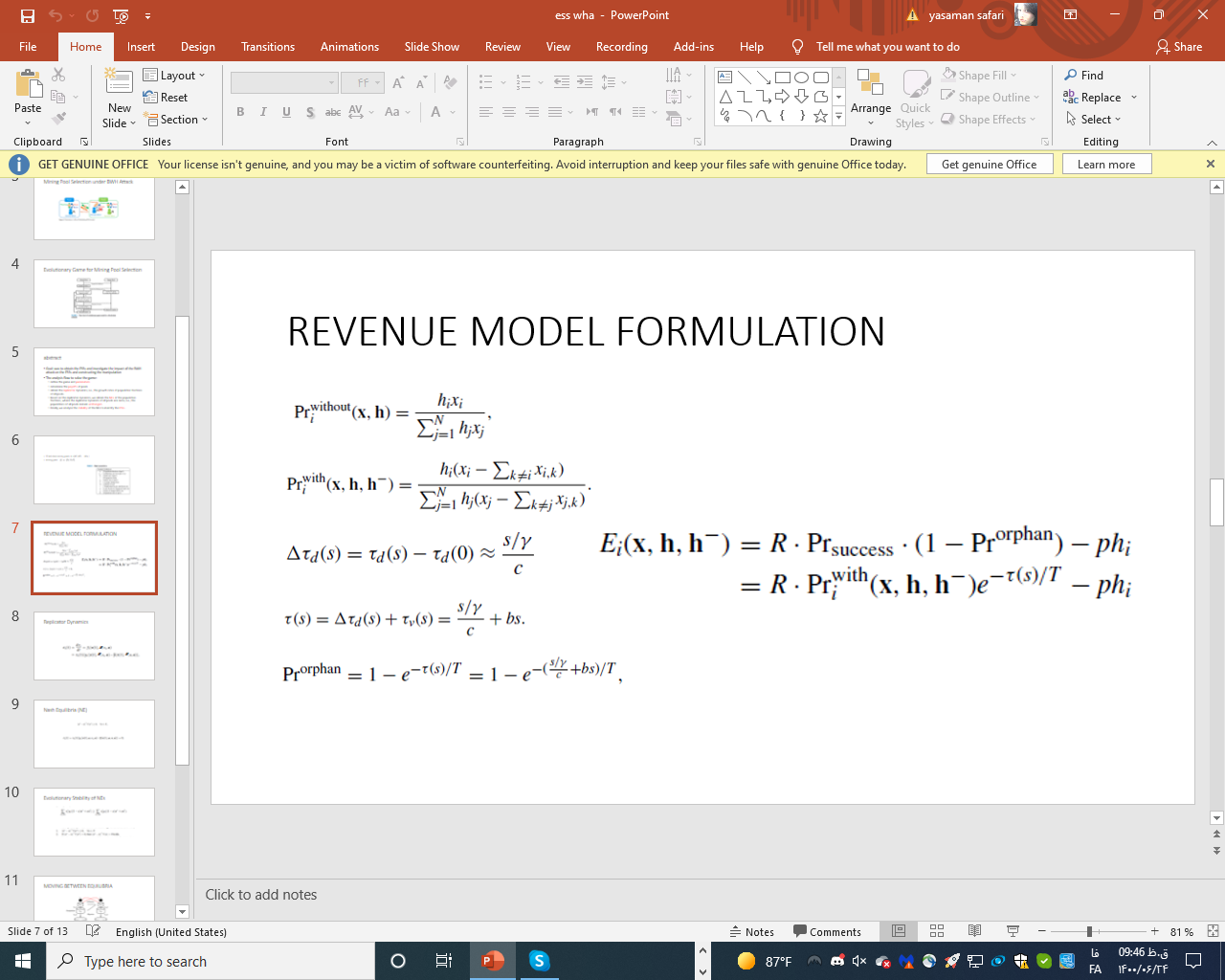
# نمودار ب دست امده باید بیانگر این باشه که نمودارها با وئجود لغزش اما سریعا برگردند به نقطه نش

معدله فیتنس مقاله که پیاده کردیم:



معادله رپلیکاتور مقاله: 

Aهمان جدول پاداش است که در این مقاله بر حسب معیارهای تعیین شده به صورت فرمول بیان شد.



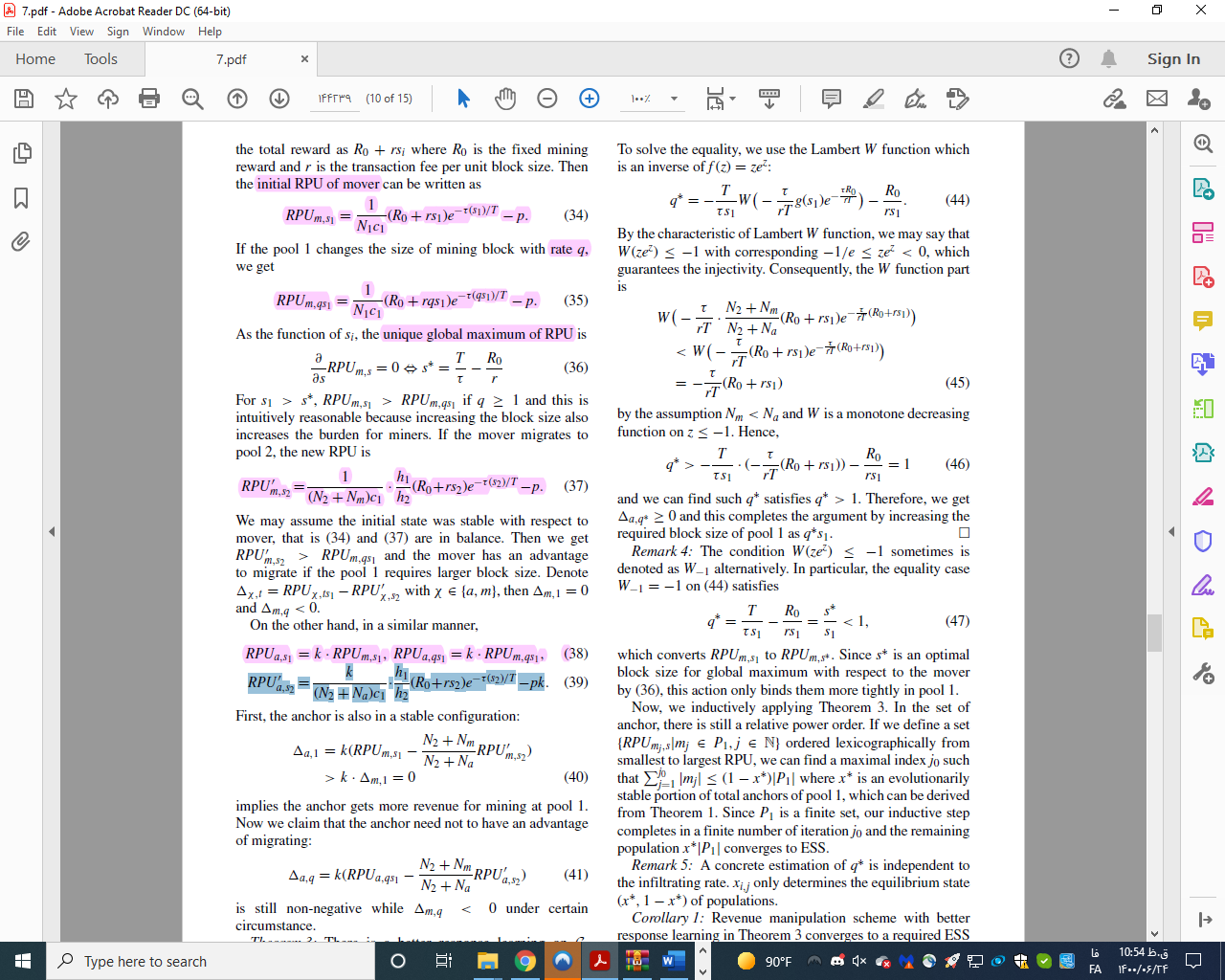
معادلات پاداش حین محاسبه معادلات فیتنس و رپلیکاتور در مقاله اعمال شدند. لذا مستقیما از همانن معادلات در پیاده سازی استفاده شد.

تا اینجا توانستیم بر اساس مقداردهی اولیه معیارها، وضعیت جمعیت ماینرهایی که داخل استخر حمله گر یا حمله کننده هستند را بررسی کنیم و نقطه تعادل سازگار انها را بیابیم. در بخش ارزیابی سه ارزیابی و تحلیل نمودارهای این بخش اورده میشود.

اما این مقاله دو بررسی دیگر نیز انجام داد. بررسی اول به این مسیله میپردازد که انگیزه ماینرها برای مهاجرت بررسی شود. در بررسی های قبلی همه چیز اشکار بود و معلوم بود کدام استخر قربانیست اما این بد است چرا که ماینرهای صادق با دیدن پاداش بهتر خرابکاری ممکن است به جمعیت انها بپویندند. لذا اینجا به انگیزه ماینرها برای مهاجرت که به پاداش بدست امده از طریق ماینینگ بلاکها وابسته است میپردازیم. در این بخش هر ماینینگ را یک متغیر تصادفی در نظر میگیریم که در تحلیلات اماری ما میتواند رویداد درنظر گرفته شود.نرخ موفقیت ماینینگ ها را متناسب با قدرت هش هر استخر محاسبه میکنیم که میتوان هر گام بازی بر حسب نرخ ماینینگ ها موفق با توزیع نرمال تخمین زد. استخر بدون ماینر نفوذی و با ماینر نفوذی هر کدام از یک توضیع نرمال پیروی میکنند. برای اینکه اطلاعات بر ماینرها اشکار نباشد لذا باید توزیع های دو استخر ب هم نزدیک باشد.فاصله باتاچاریا توزیع ها محاسبه شد و مقدار اندکی بود که نشان از ان است که ماینر ها نمیتوانند از وجود حمله خارجی باخبر باشند. لذا بدون در نظر گرفتن این مسیله تصمیم ب مهاجرت باید بگیرند. برای جداسازی تو دسته از یک جداساز خطی استفاده شد که سعی میکند دو کلاس را بنحوی ک میانگینها در دورترین فاصله از هم و واریانسها به کمترین مقدار برسد جدا کند.

از آنجایی که این بخش ارتباط خیلی کمتری با اصل مقاله یعنی نظریه بازی ها داشت و برای پیاده سازی مقادیر دقیق برخی پارامترها وجود نداشت و نیاز به پیاده سازی دو توزیع نرمال و نمونه گیری پواسون داشت صرف نظر کردم و به همین فهم کلی بسنده کردم.

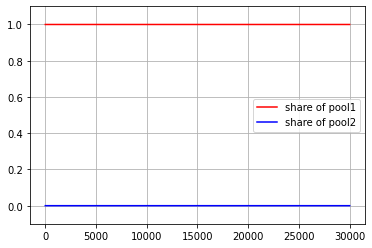
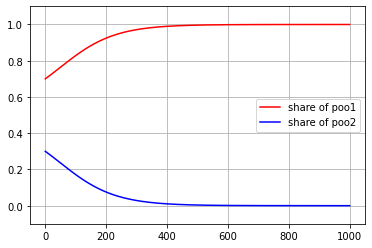
اما در بخش آخر معیار سایز بلاک به عنوان یکی از معیار های تاثیرگذار در تعیین نقاط ess بررسی شد. برای این منظور فرض شد که سایز بلاک بتواند با یک ضریبی تغییر کند. هر چه سایز کوچکتر، محاسبات سریعتر و هزینه کمتر خواهد داشت. ماینر های یک استخر به عنوان مهاجر یا انچر در نظر گرفتتیم. کسی که مهاجرت میکند طبیعتا قدرت هش کمتری دارد که ترجیح میدهد برود دزدی! اما کسی که میماند قدرت هس بیشتری دارد. برای این بخش یک معیار جدید به اسم RPU معرفی شد که پاداش در واحد بلاکها با سایز S را معرفی کند. این معیار وابسته به پاداش ماینینگ و نرخ هش و نرخ حمله و سایت بلاک و ... است. معادلات مربوط به هر ماینر متعلق به انچر یا مهاجر با سایز بلاک فعلی، سایز بلاک تغییر یافته با ضریب Q و نهایتا در حالتیکه با سایز ثابت مهاجرت کند، در مقاله امده است. نمودارهای مربوطه در بخش ارزیابی رسم و بررسی خواهند شد.

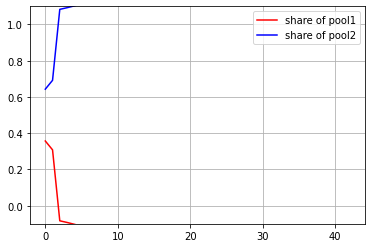
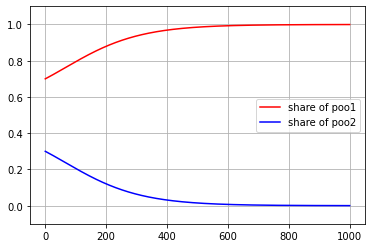


**ارزیابی**

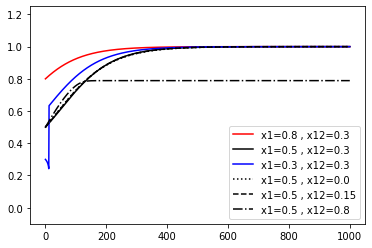
بخش1: ESS

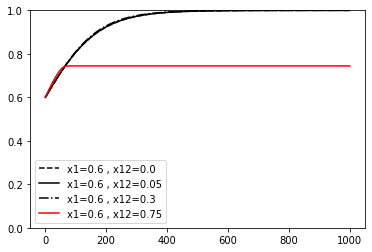
ابتدا در مساله به دنبال یک نقطه ای از جمعیت هر استخر به عنوان نقطه پایدار بودیم.نمودار زیر جمعیت هر استخر را با مقادیر اولیه ای که در بخش پیاده سازی بیان شدمدل کرده است.همانطور که دیده میشود جمعیت استخر اول به سمت همه گیر شدن میرود. استخر اول در اینجا اسم یک استخر خاص نیست بلکه منظور استراتژی با ویژگی های معین است. در اینجا چون میزان پاداش ماینینگ بالا بود و نرخ های هش کم بود، جمعیت ترجیح داد تا از طریق نفوذ پاداش خود را بیشینه کند. در عکس بعدی نتیجه بررسی وضعیت جمعیت ها بعد از دستکاری اپسیلونی نقطه نش دیده میشود. همانطور که میبینید همچنان جمعیت ها در 1و0 پایدارند. اما در شکل سوم برای نقطه دیگری که نش پایدار نیست تست شد و نمودار جهش پیئدا کرد و به نش برنگشت پس ان نقطه نش پایدار نیست.



سپس سعی شد تا نمودار بعدی کتاب رسم شود که همان حالت اول با تغییراتی در مقادیر پارامترهاست. اینبار با افزایش چشمگیرتر پاداش ماینینگ و افزایش نرخ هش ها سعی شد تا نش پایدار داخلی بین صفر و یک یافت شود. اما به دلیلی که واقعا نمیدانم نتایج نمودارها همانند مقاله نیست. شکل یک باید نقطه برخورد میبود در حالیکه نرخ رشد ها دقیقا برعکس است. اینجا باید یک نقطه میانی که مقاله گفته پایدار میشد که طبق نمودار سومی رخ نداد چون ثابت نشده.

و نهایتا برای بررسی نقش قدرت هش و نرخ حمله در جمعیت و ESS نمودارهای زیر رسم شد. در این حالت هر بار یک معیار ثابت و بقیه تغییر کردند. در نمودار اول، میبینید که جمعیت های با نرخ اولیه متفاوت اما نهایتا به نش همگرا شدند. دلیل این همگرایی افزایش میزان پاداش و قدرت هش است که ماینرها را به سمت نفوذی بودن جلب میکند. سه نمودار مشکلی در نمودار اول قرار است همان نتایج نمودار دوم را بیان کنند. همانطور که ملاحظه میکنید، همگی جز یکی به سمت نش پایدار که استخر یک یا حمله غالب باشد همگرا میشوند. ان مورد که مثل بقیه همگرا نمیشود نرخ حمله به شدت بالاتری دارد. همانطور ک دیدید تا یک جایی افزایش نرخ حمله پایداری را افزایش میدهد اما از یک جا ب بعد برعکس عمل میکند. از این مسیله برای تنظیم نرخ حمله میتوان استفاده کرد.

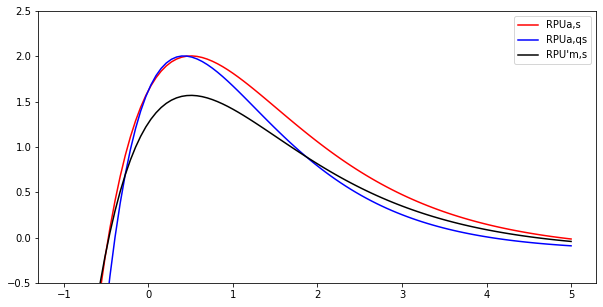
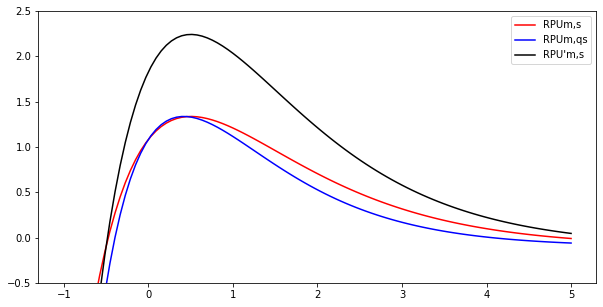




بخش2: RPU

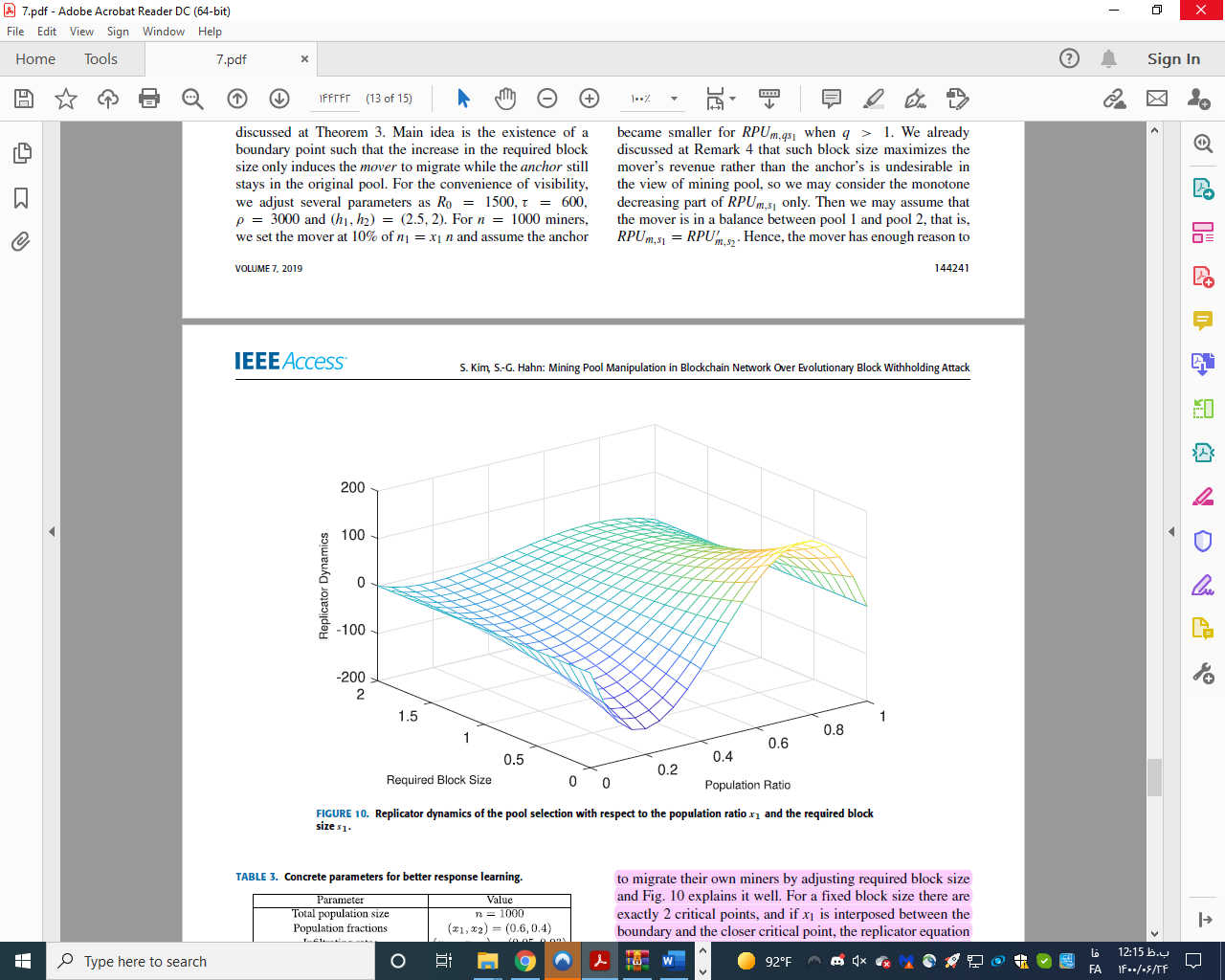
در نهایت سعی میکنیم با تنظیم سایز بلاک مناسب مینیمم که هزینه کمتری هم بخواهد به نش برسیم و سایز بلاک را تعیین کنیم. معادلات مطرح شده در بخش قبلی را پیاده سازی کردیم و نمودارهای زیر به دست آمد. نمودار او.لی متعلق به ماینرهاییست که مهاجرت میکنند. همانگونه که میبینید سایز بلاک برای ماکزیمم پاداش برای موور داخل استخر اول قبل و بعد افزایش سایز بلاک تقریبا یکسانه اما با افزایش سایز بلاک پاداشش کاهش میابد. اما هنمانطور ک میبینید میزان پاداش بلاک افزایش یافته با میزان پاداش مهاجرت به استخر دوم در یک خط افقی برابر است و این میتواند ماینر را به مهاجرت بیاندازد.

در شکل دوم وضعیت انچرهاست که قدرت محاسباتی بالاتری دارند و تا یک جایی حتی با افزایش سایز بلاک هم ترجیح میدهند مهاجرت نکنند که خب این نقطه تعیین سایز بلاک ما خواهد بود.



**نتیجه نهایی:**

همانطور که تا الان دیدید و در شکل زیر میبینید، نمیتوان بهترین پاداش را هیچ استخر یا ماینری داد بلکه تعادل نش جایی مابین است چیزی مثل معمای زندانی. ولی میتوان با تنظیم مقادیر اولیه و معیارها در مقادیر مناسب، به تعادل نش بهتری رسید.



**منابع**:

برای پیاده سازی از سایت زیر کمک گرفته شد

<http://systems-sciences.uni-graz.at/etextbook/gametheory/replicator.html>