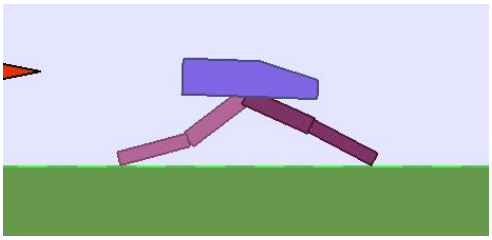
Испит из Основних оптимизационих алгоритама у инжењерству

Индекс	Презиме и име	Датум	Поени	Испит	Оцена
2018/3143	Рошко Бојан	23.1.2019.			

1. Опис проблема оптимизације

Веб платформа https://gym.openai.com/ нуди неколико окружења (у виду различитих проблема) за учење и тестирање алгоритама за машинско учење са повратном спрегом (reinforcement learning). За овај пројекат је коришћено окружење BiPedalWalker, које за циљ има да истренира кретање јако поједностављеног дводимензионалног робота са 4 зглоба.



Платформа сама по себи нуди аутоматску обзервацију и аутоматско додељивање скора одређеним акцијама. Циљ оптимизације је био добити сто већи скор на овој платформи, сто је значило боље (брже) кретање. Робот се контролише са 4 параметара, који имају вредност између -1 и 1 и одређују колика је снага на мотору на сваком од зглобова. Оно што заправо генетски алгоритам одређује је линеарна зависност обзервација и акција. Како сви ови параметри могу имати реалне вредности, овај проблем спада у НЛП проблеме.

2. Дефиниција оптимизационе функције и оптимизационог простора

Обсервације добијене од платформе садрже 24 варијабли (14 који говоре о угловима зглобова, углу робота, брзини зглобова, брзини робота и рељефу терена). У сваком тренутку је потребно одговорити са 4 вредности који одговарају снази на мотору на сваком од зглобова ногу. У овом пројекту је испитивана могучност једноставног

контролисања помоћу линеарне зависности контроле мотора од пристиглих обсервација. Ово значи да је овај проблем рађен на 24*4 = 96 димензионалном простору. Овај проблем има један критеријум (али који нама није познат, што је и поента примене машинског учења са повратном спрегом). Оно што нам платформа враћа као вредност оптимизационе функције је вредност која зависи од величине помака робота, да ли је робот пао, да ли је робот стао и правилности кретања.

3. Оптимизациони алгоритам

За оптимизацију овог проблема је коришћен генетски алгоритам. Генетски алгоритам се обично примењује над проблемима већих димензионалности (као што је овај), и који немају неких очигледних правилности. Генетски алгоритам је примењен тако да се од претходне генерације изабере одређени број јединки које опстају, а онда се све јединке заједно укрштају вероватноћом која одговара њиховом фитнесу до поновне исте величине нове генерације. Такође, при укрштању се примењују технике кросовера и мутација. Параметри су благо варирани ради испитивања брзине конвергенције решења и квалитета добијања крајњег решења.

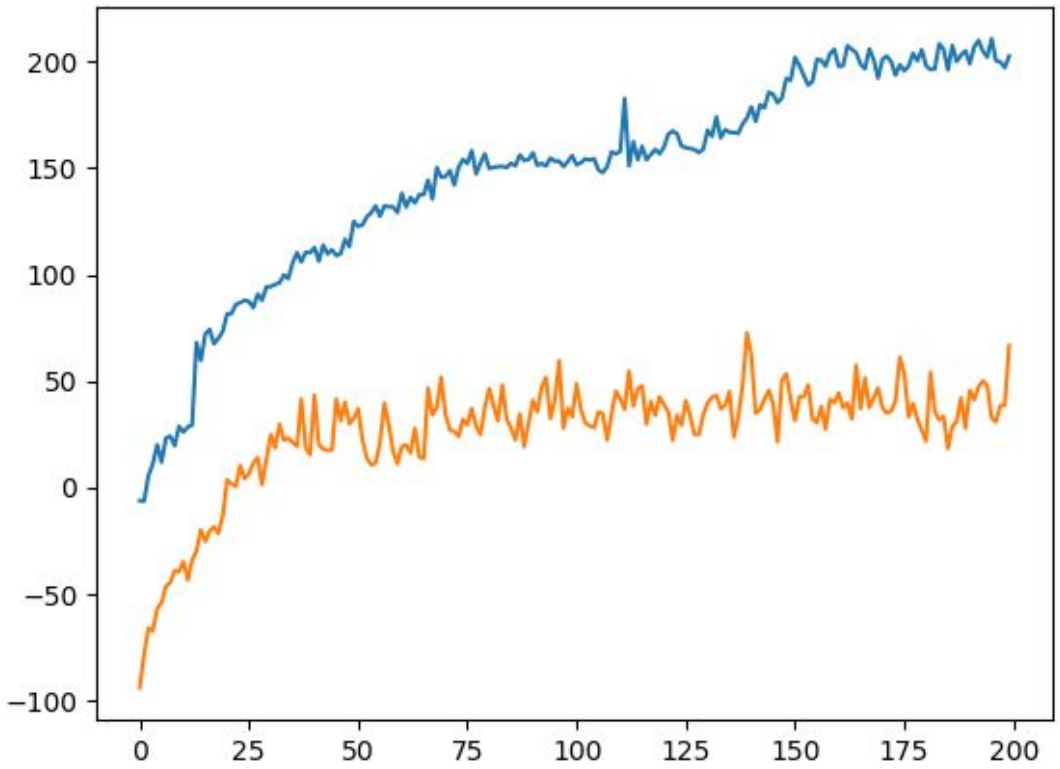
4. Прилози

Уз пројекат су достављени Python фајлови који су коришћени са оптимизацију (тренирање), и за визуелизацију и приказ решења.

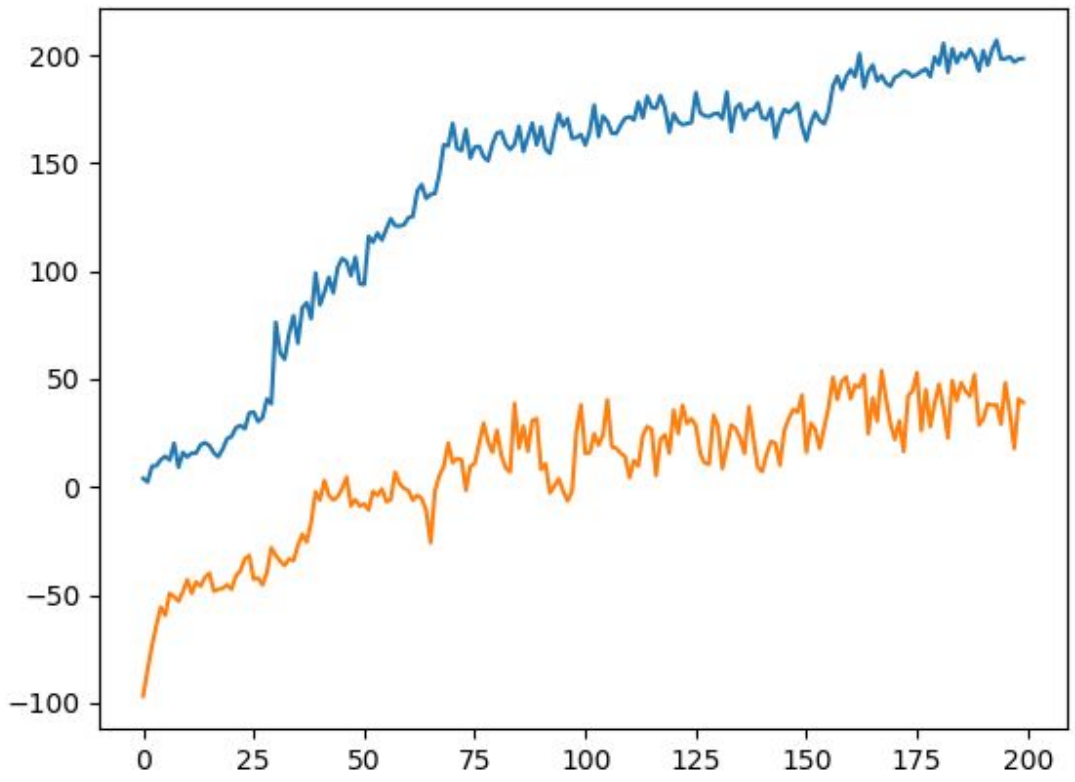
5. Резултати

У наставку се налазе наведени параметри генетског алгоритма и пратећи графици који показују промену најбољег фитнеса и средњег фитнеса генерације. На крају се налази и опис обсервационих варијабли и акционих варијабли које су коришћене у платформи.

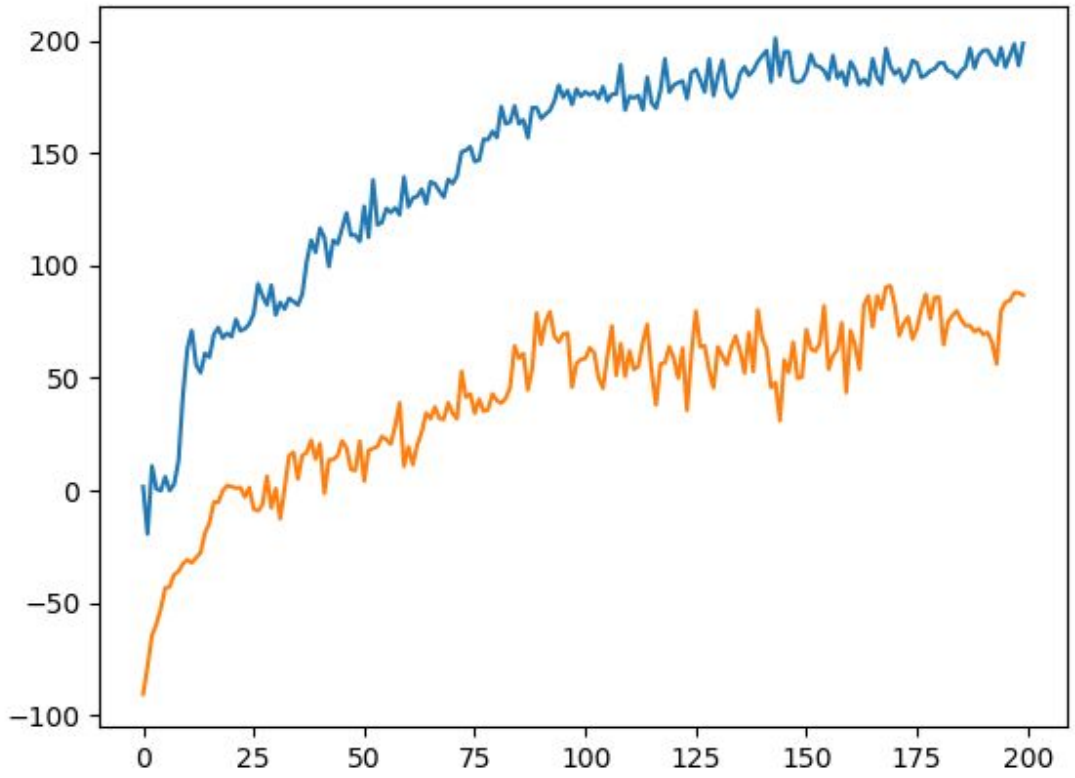
cross: 0.98,
mutation: 0.02,
select: 10,
num_iter: 1000,
gen_size: 100,
num_generations: 200



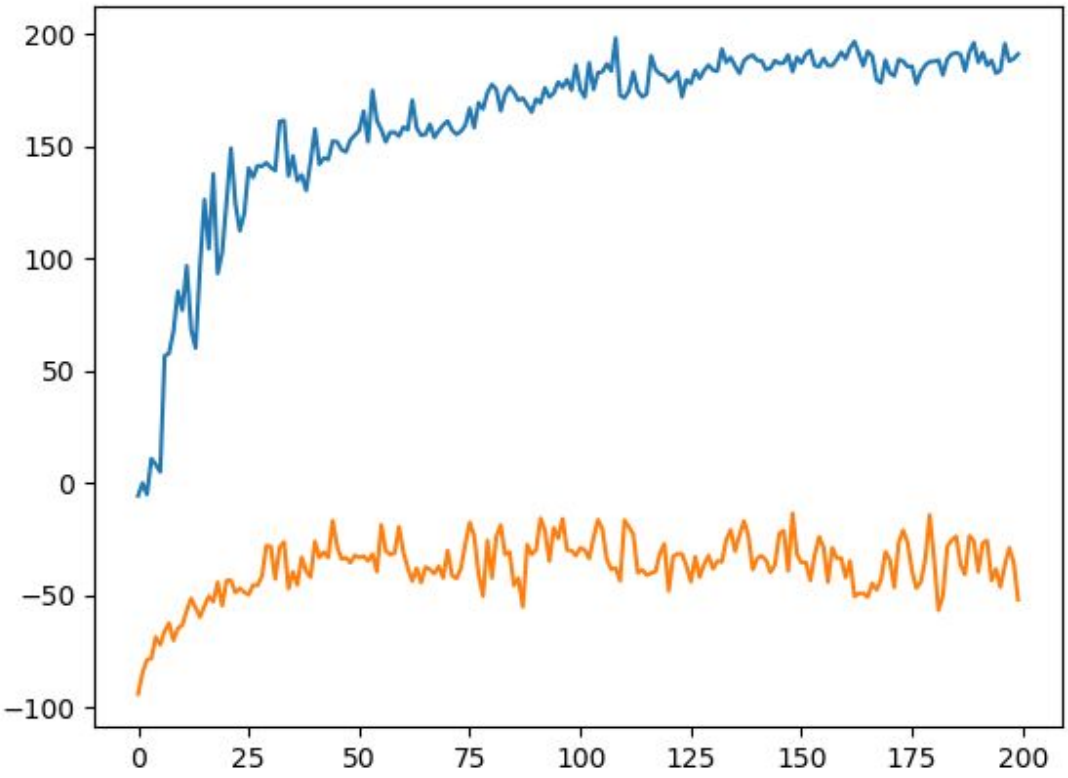
cross: 0.98,
mutation: 0.02,
select: 3,
num_iter: 1000,
gen_size: 100,
num_generations: 200



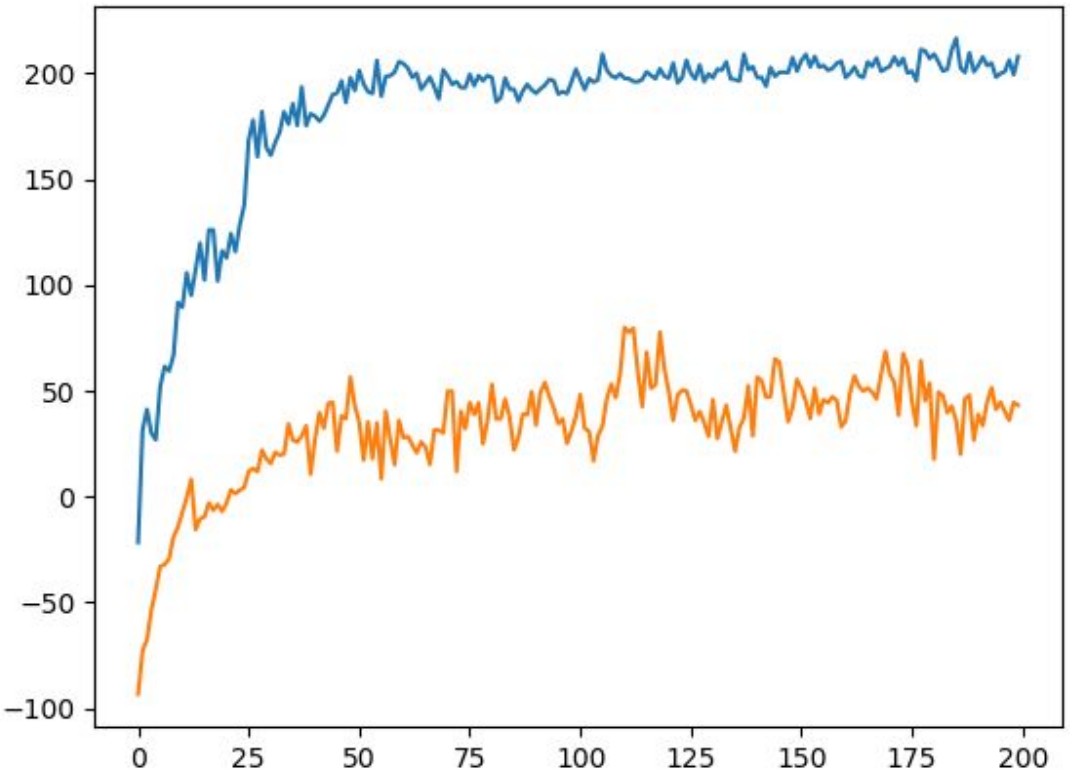
cross: 0.98,
mutation: 0.02,
select: 30,
num_iter: 1000,
gen_size: 100,
num_generations: 200



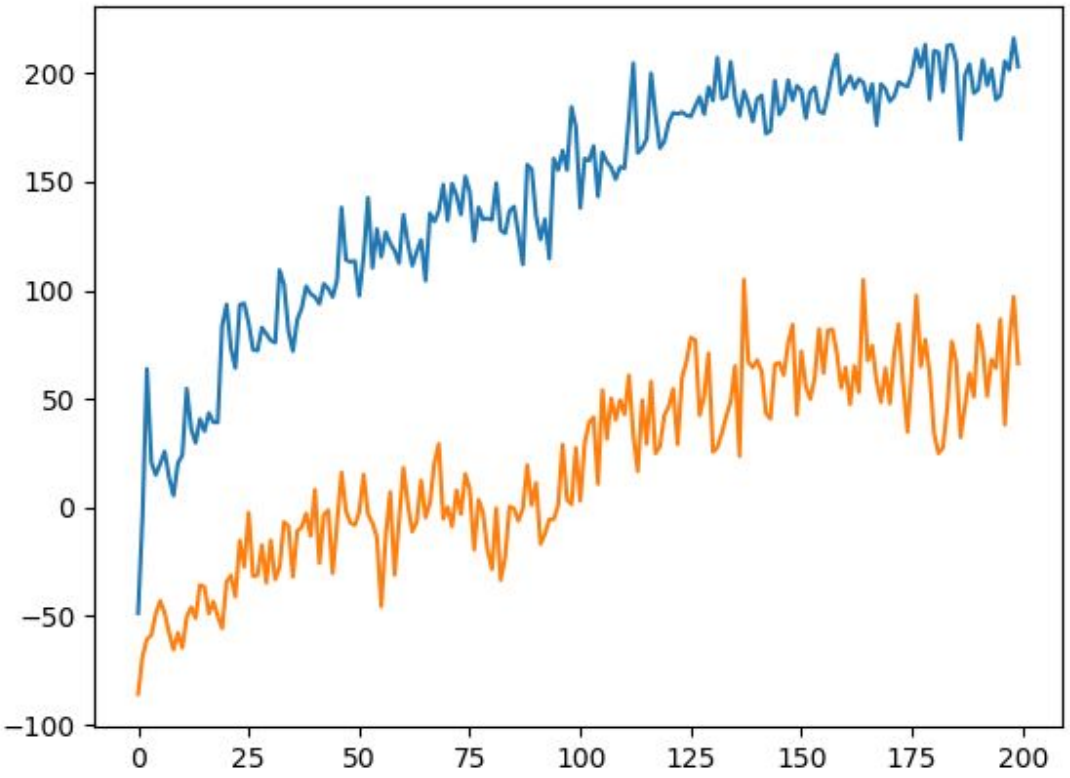
cross: 0.98,
mutation: 0.1,
select: 10,
num_iter: 1000,
gen_size: 100,
num_generations: 200



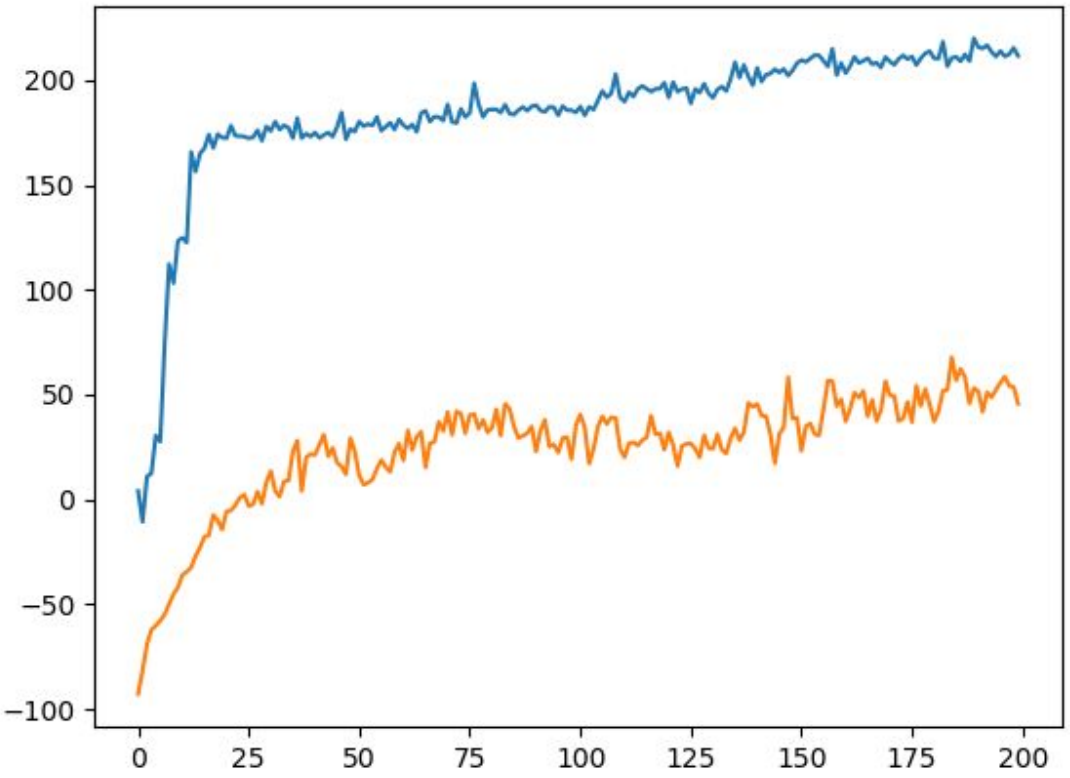
cross: 0.85,
mutation: 0.02,
select: 10,
num_iter: 1000,
gen_size: 100,
num_generations: 200



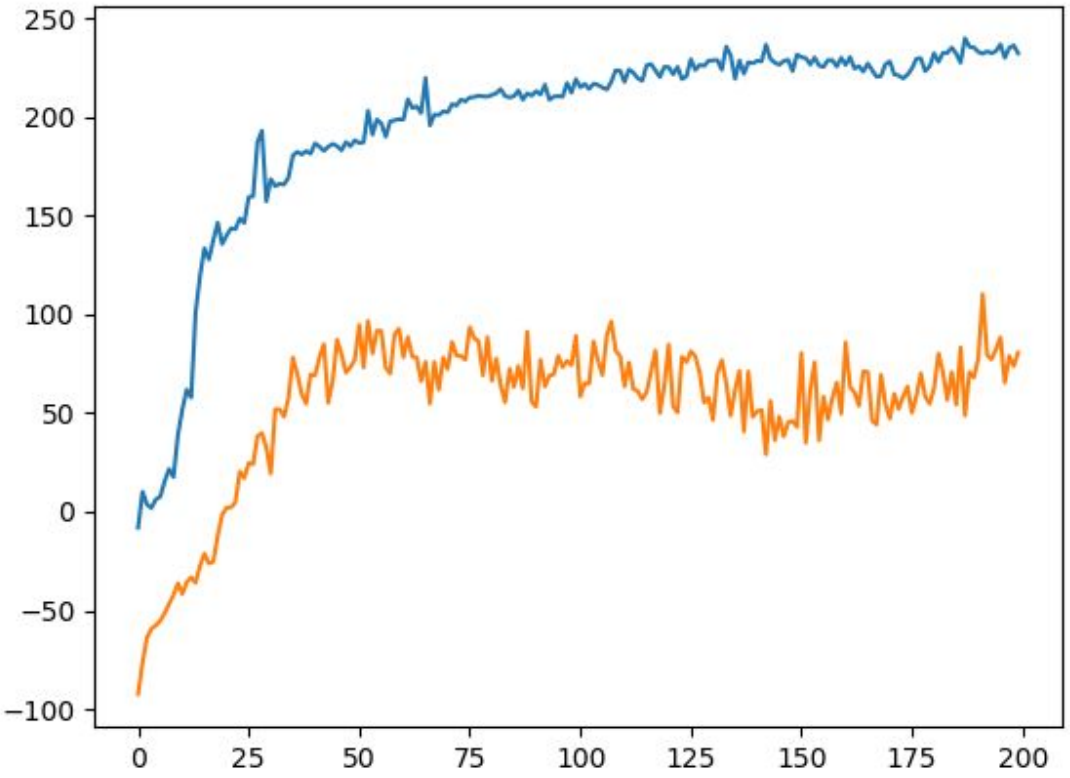
cross: 0.98,
mutation: 0.02,
select: 10,
num_iter: 1000,
gen_size: 30,
num_generations: 200



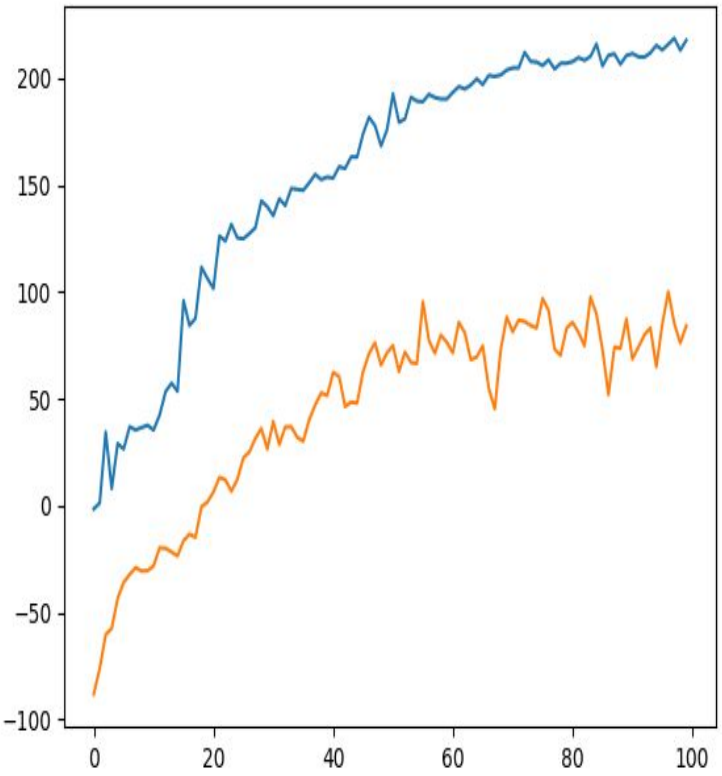
cross: 0.98,
mutation: 0.02,
select: 10,
num_iter: 1000,
gen_size: 300,
num_generations: 200



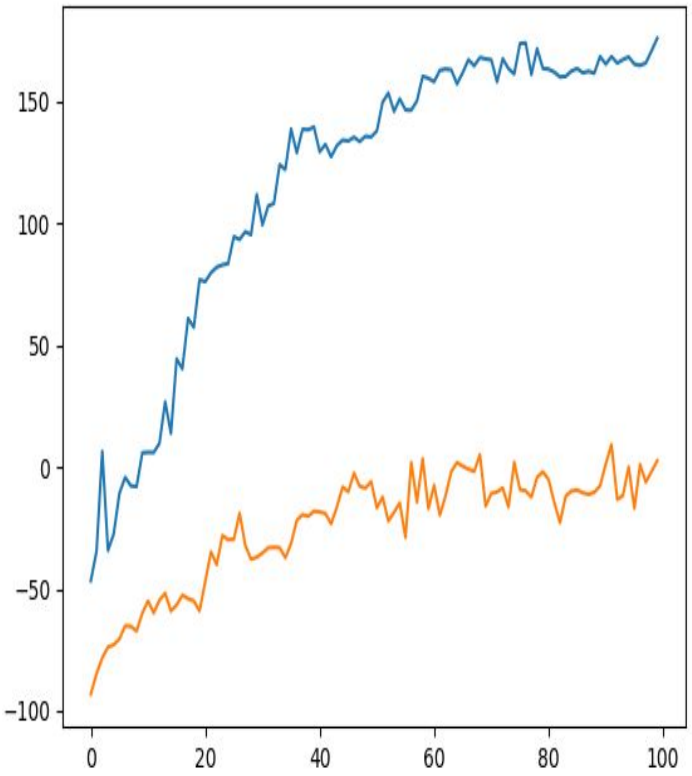
cross: 0.98,
mutation: 0.02,
select: 10,
num_iter: 3000,
gen_size: 100,
num_generations: 200



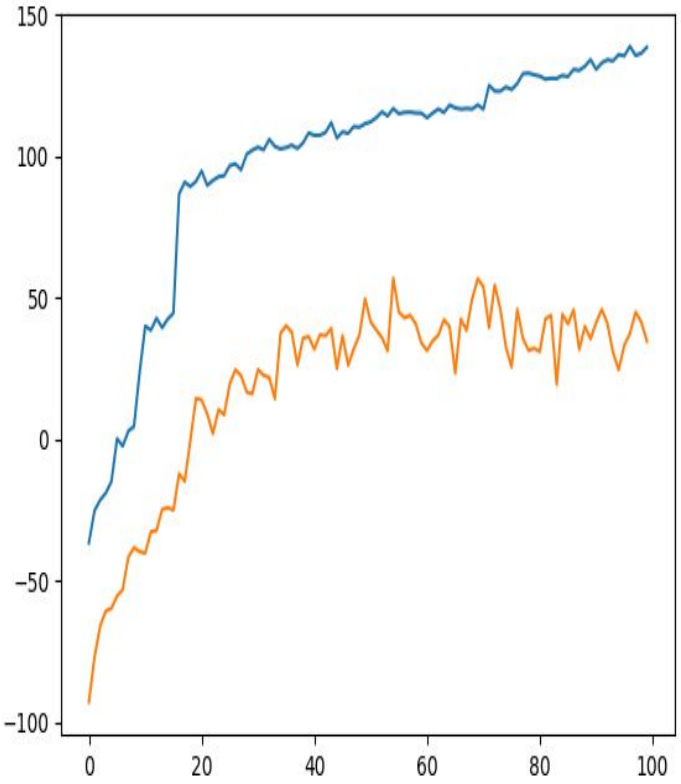
cross: 0.98
mutation: 0.02
select: 10
num_iter: 1000
gen_size: 100
num_generations: 100
num_repeat: 5



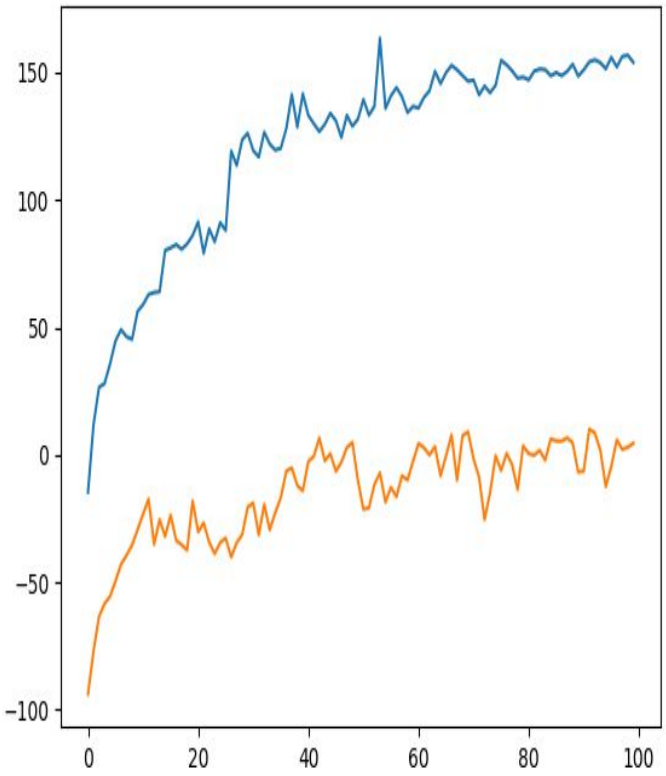
cross: 0.85
mutation: 0.08
select: 10
num_iter: 1000
gen_size: 100
num_generations: 100
num_repeat: 5



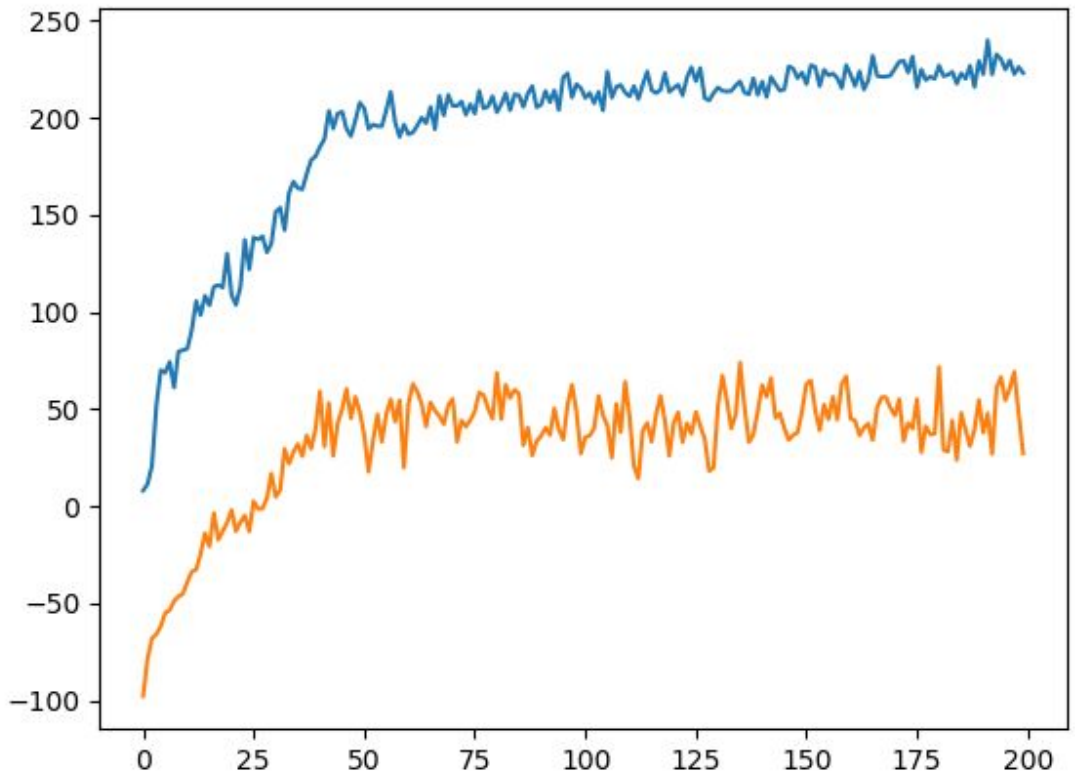
cross: 0.98
mutation: 0.02
select: 10
num_iter: 1000
gen_size: 100
num_generations: 100
num_repeat: 20



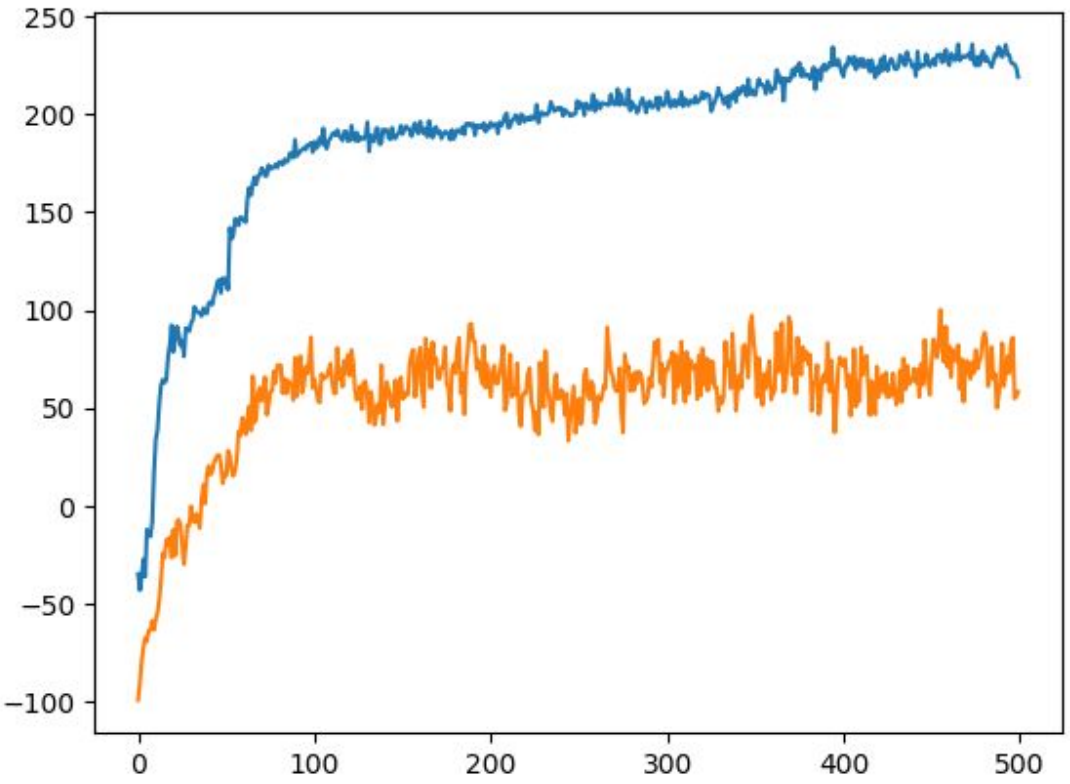
cross: 0.98
mutation: 0.02
select: 10
num_iter: 1000
gen_size: 100
num_generations: 100
num_repeat: 3



cross: 0.98
mutation: 0.02
select: 10
num_iter: 1000
gen_size: 100
num_generations: 200
num_repeat: 1
zeros_lidar



cross: 0.95
mutation: 0.02
select: 10
num_iter: 1000
gen_size: 100
num_generations: 500
num_repeat: 3
zeros_lidar



Observation

Type: Box(24)

Num	Observation	Min	Max	Mean
	hull_angle	0	2*pi	0.5
1	hull_angularVelocity	-inf	+inf	-
2	vel_x	-1	+1	-
3	vel_y	-1	+1	-
4	hip_joint_1_angle	-inf	+inf	-
5	hip_joint_1_speed	-inf	+inf	-
6	knee_joint_1_angle	-inf	+inf	-
7	knee_joint_1_speed	-inf	+inf	-
8	leg_1_ground_contact_flag	0	1	-
9	hip_joint_2_angle	-inf	+inf	-
10	hip_joint_2_speed	-inf	+inf	-
11	knee_joint_2_angle	-inf	+inf	-
12	knee_joint_2_speed	-inf	+inf	-
13	leg_2_ground_contact_flag	0	1	-
14-23	10 lidar readings	-inf	+inf	-

Actions

Type: Box(4) - Torque control(default) / Velocity control - Change inside /envs/box2d/bipedal_walker.py line 363

Num	Name	Min	Max
	Hip_1 (Torque / Velocity)	-1	+1

1	Knee_1 (Torque / Velocity)	-1	+1
2	Hip_2 (Torque / Velocity)	-1	+1
3	Knee_2 (Torque / Velocity)	-1	+1