Inżynieria wiedzy i symboliczne uczenie maszynowe

Sprawozdanie z projektu

Uczenie ze wzmocnieniem w środowisku Robocode

Patryk Czuchnowski

Kod źródłowy: https://github.com/patczuch/iwisum-robocode

Wprowadzenie

W ramach projektu stworzyłem robota działającego w środowisku Robocode. Robocode to edukacyjne środowisko symulacyjne służące do nauki programowania oraz testowania algorytmów poprzez tworzenie autonomicznych, walczących ze sobą czołgów (robotów). Programista definiuje logikę działania robota, a następnie są one uruchamiane na wirtualnym polu bitwy, gdzie rywalizują z innymi jednostkami.

Przykładowa bitwa w środowisku Robocode

W projekcie skupiłem się na zastosowaniu metody uczenia ze wzmocnieniem, aby umożliwić robotowi doskonalenie swoich działań na podstawie informacji zbieranych z otoczenia.

Projekt został zrealizowany w języku Java przy użyciu biblioteki Robocode.

Algorytm

W projekcie zastosowano algorytm Q-learning, który pozwala na uczenie się optymalnej strategii działania w środowisku na podstawie interakcji agenta z otoczeniem. Wartość funkcji Q była aktualizowana na podstawie otrzymywanych nagród, a decyzje były podejmowane z uwzględnieniem eksploracji i eksploatacji strategii.

W celu poprawy efektywności uczenia, zastosowałem mechanizm malejącego współczynnika eksploracji. Dzięki temu początkowo agent częściej podejmował losowe decyzje, natomiast w miarę zdobywania doświadczenia skłaniał się bardziej ku sprawdzonym działaniom.

Głównymi celami nauki były:

- Trzymanie przeciwnika na celowniku
- Utrzymywanie przeciwnika w polu widzenia
- Strzał gdy przeciwnik jest na linii strzału
- Unikanie strzału przeciwnika
- Wygranie bitwy

Obserwacje

Stan środowiska został zdyskretyzowany, aby umożliwić jego efektywne przetwarzanie przez algorytm Q-learning. W skład obserwacji wchodziły następujące cechy:

- Dystans do przeciwnika – podzielony na 5 bucketów,
- Poziom energii własnej – 3 buckety energii,
- Rotacja korpusu – 6 bucketów kątowych,
- Potrzebna rotacja działa, aby wycelować w przeciwnika – 10 bucketów - jednym z celów robota było zminimalizowanie tej wartości do zera,
- Rotacja działa przeciwnika – 6 bucketów,
- Wykrycie utraty energii przeciwnika – zmienna logiczna (bool); interpretowana jako możliwy strzał, co miało uczyć robota unikania ostrzału poprzez ruch,
- Czy przeciwnik znajduje się na linii strzału – dodatkowa flaga (bool), uwzględniająca odchylenie do 3 stopni, pomagająca w nauce celnego strzelania.

Akcje

Zbiór możliwych działań robota również został zdyskretyzowany. Agent mógł wykonywać następujące akcje:

- Ruch przód/tył – 10 poziomów prędkości (co 5 jednostek),
- Obrót ciała/działa w lewo lub prawo – 9 poziomów (co 5 stopni),
- Strzał – moc od 1 do 3 z krokiem co 0.5.

Podczas każdej decyzji istniała określona szansa na podjęcie losowej akcji, zgodnie z wartością współczynnika eksploracji. W celu poprawienia zdolności robota do nauki strzelania, jeśli przeciwnik znajdował się na linii strzału, to szansa na wybranie akcji strzału była zwiększona do 50%.

Nagrody

Robot otrzymywał nagrody za następujące działania:

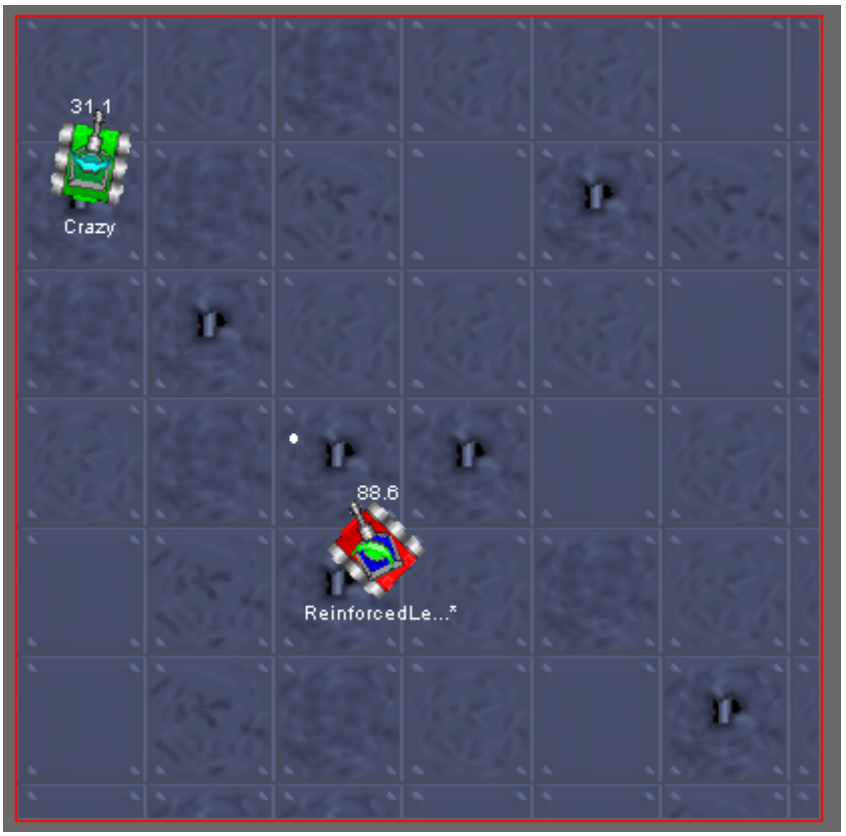
- Nagrodę za zadanie obrażeń przeciwnikowi,
- Karę za otrzymanie obrażeń,
- Dużą nagrodę za wygraną rundę,
- Karę za przegraną,
- Małą nagrodę za utrzymywanie przeciwnika w zasięgu widzenia,
- Karę za brak widoczności przeciwnika,
- Dodatkową nagrodę za mały kąt konieczny do obrotu działa w stronę przeciwnika – im mniejszy, tym większa nagroda,
- Karę za zbyt dużą liczbę rund – miało to zachęcać do szybszego rozstrzygania walk.

Trening

Bitwy były rozgrywane na planszy 400x400. Trening przeprowadziłem przeciwko wbudowanemu Crazy Robotowi - uznałem że przez jego losowe ruchy mój robot będzie w stanie się nauczyć jak reagować w wielu sytuacjach.

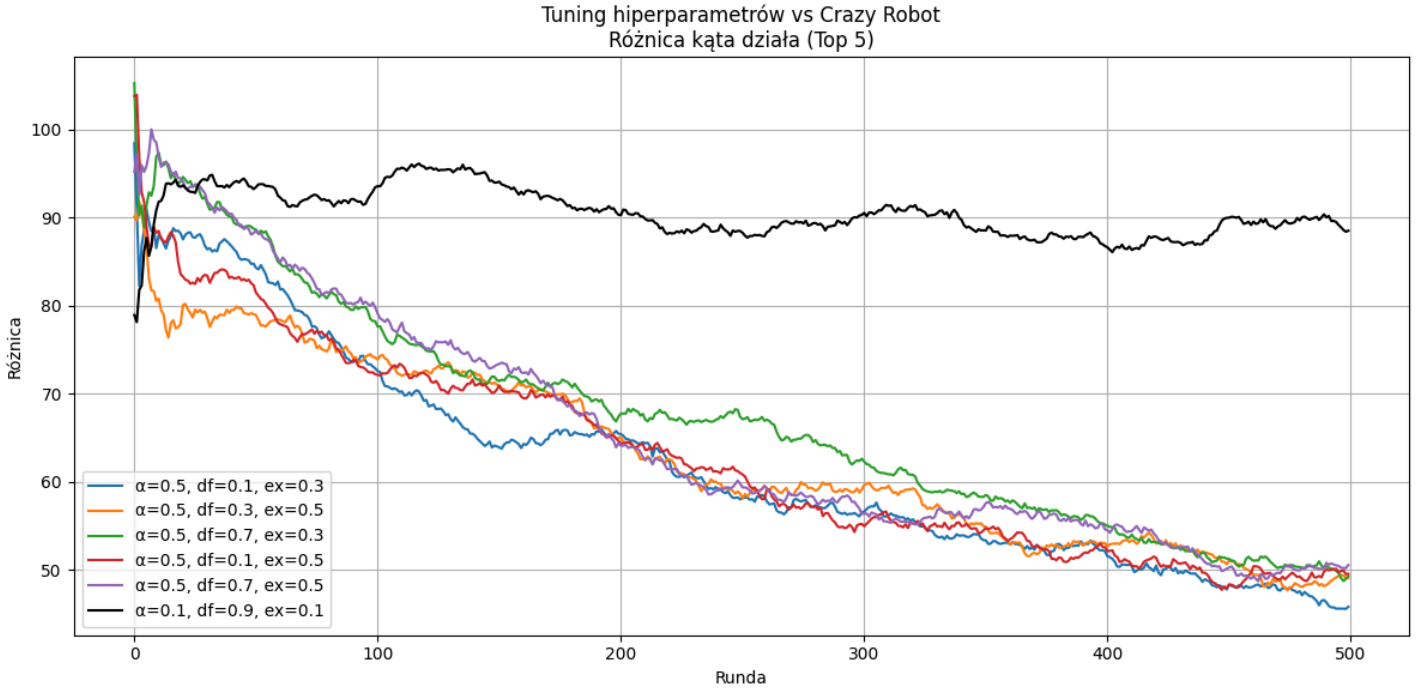
W celu przyspieszenia treningu i poprawy jakości robota przeprowadziłem tuning hiperparametrów przy pomocy GridSearch. Przeprowadziłem po 500 bitew dla każdego zestawu. Przetestowałem iloczyn kartezjański następujących wartości:

- alpha $\in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
- discountFactor $\in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
- experimentRate $\in \{0.1, 0.3, 0.5, 0.7\}$

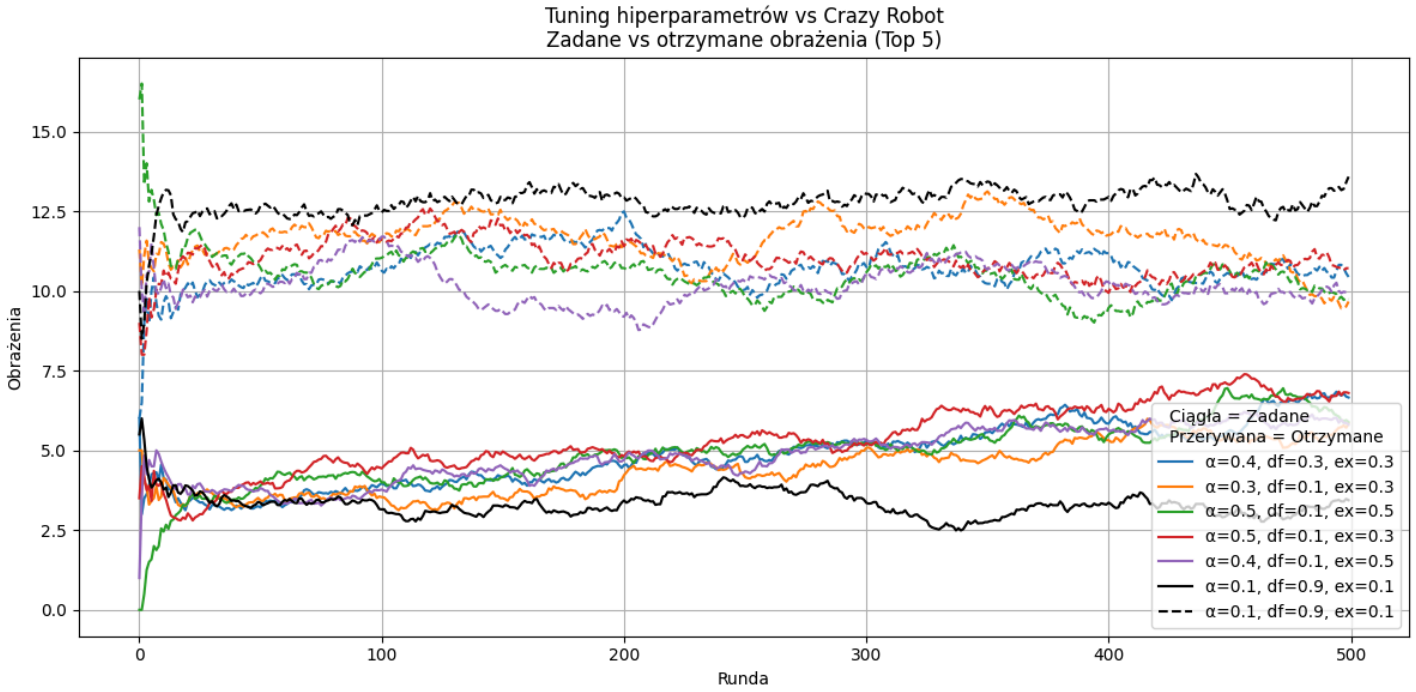


Robot w akcji

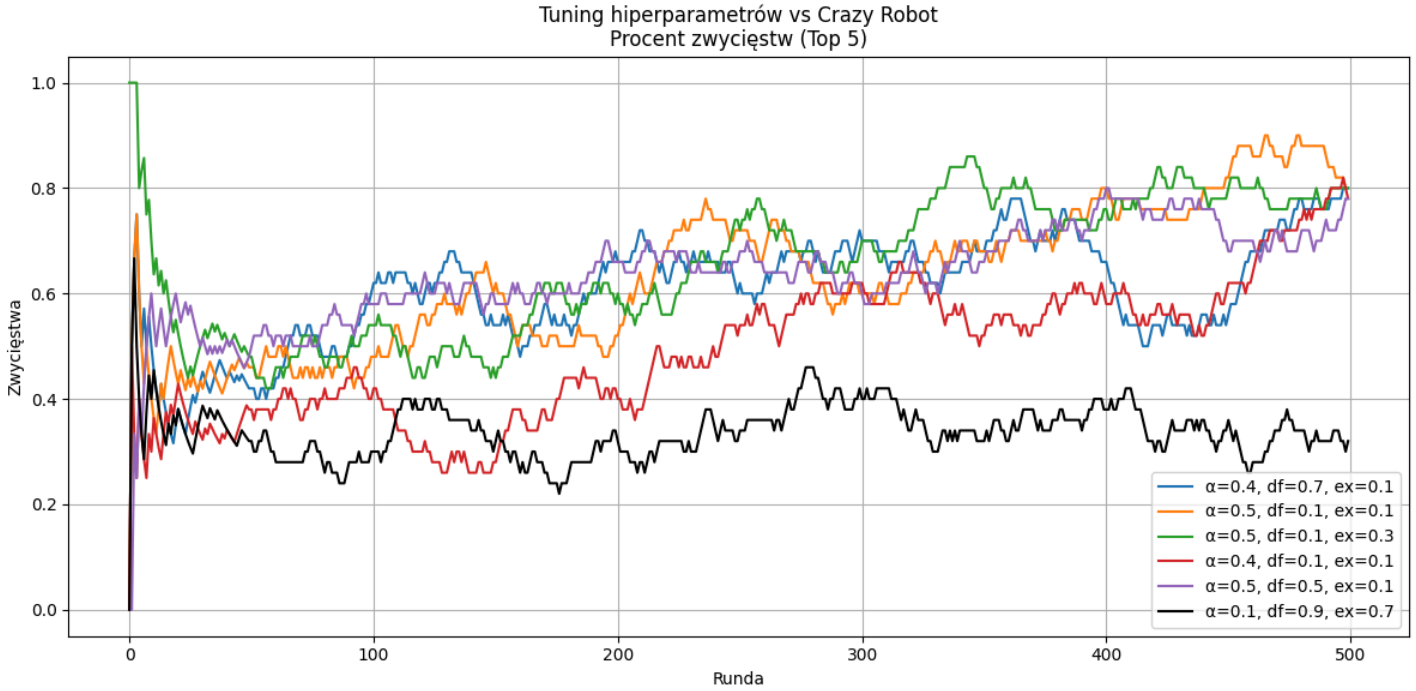
Stworzyłem wykresy najlepszych zestawów hiperparametrów dla 3 metryk - potrzebnego obrotu działa do przeciwnika, zadanych i otrzymanych obrażeń oraz procentu zwycięstw. Metryki były liczone dla 10 ostatnich bitew. Należy pamiętać, że na wykresach jest zaprezentowane jedynie 5 najlepszych zestawów - w przypadku wyświetlenia wszystkich 100 wykresy były zupełnie nieczytelne. Dla kontrastu czarnym kolorem zaznaczyłem najgorzej radzący sobie zestaw.



Najlepiej radzą sobie zestawy parametrów z dużą alfą - zapewne żeby te z niższymi wartościami były lepsze potrzebowalibyśmy większej liczby rund.



Zadawane obrażenia rosną, a otrzymywane maleją, nadal jednak robot więcej dostaje niż zadaje.



Procent zwycięstw osiąga zadowalający poziom 80%.

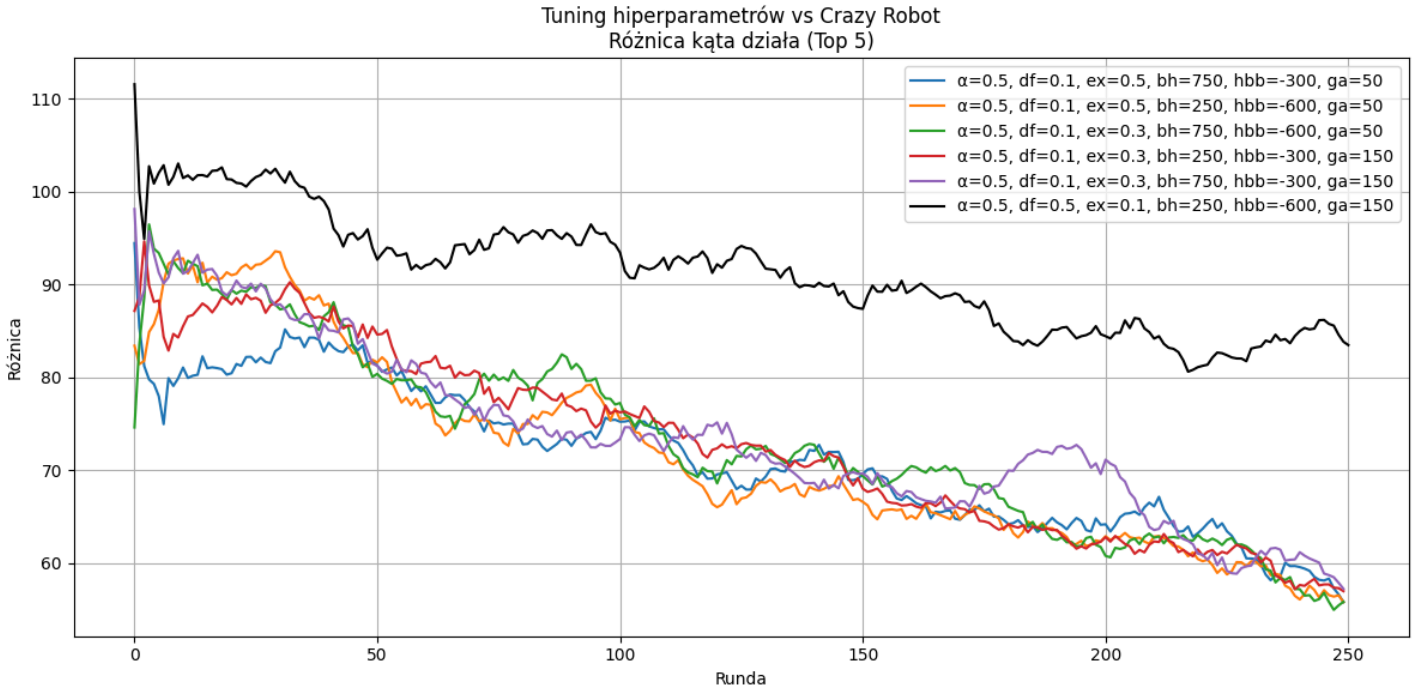
Na podstawie tych wykresów zidentyfikowałem 10 zestawów hiperparametrów które poradziły sobie najlepiej:

alpha	discountFactor	experimentRate
0.5	0.1	0.5
0.5	0.1	0.3
0.5	0.1	0.1
0.5	0.3	0.5
0.5	0.5	0.1
0.5	0.7	0.3
0.5	0.7	0.5
0.4	0.3	0.3
0.4	0.1	0.5
0.4	0.1	0.1

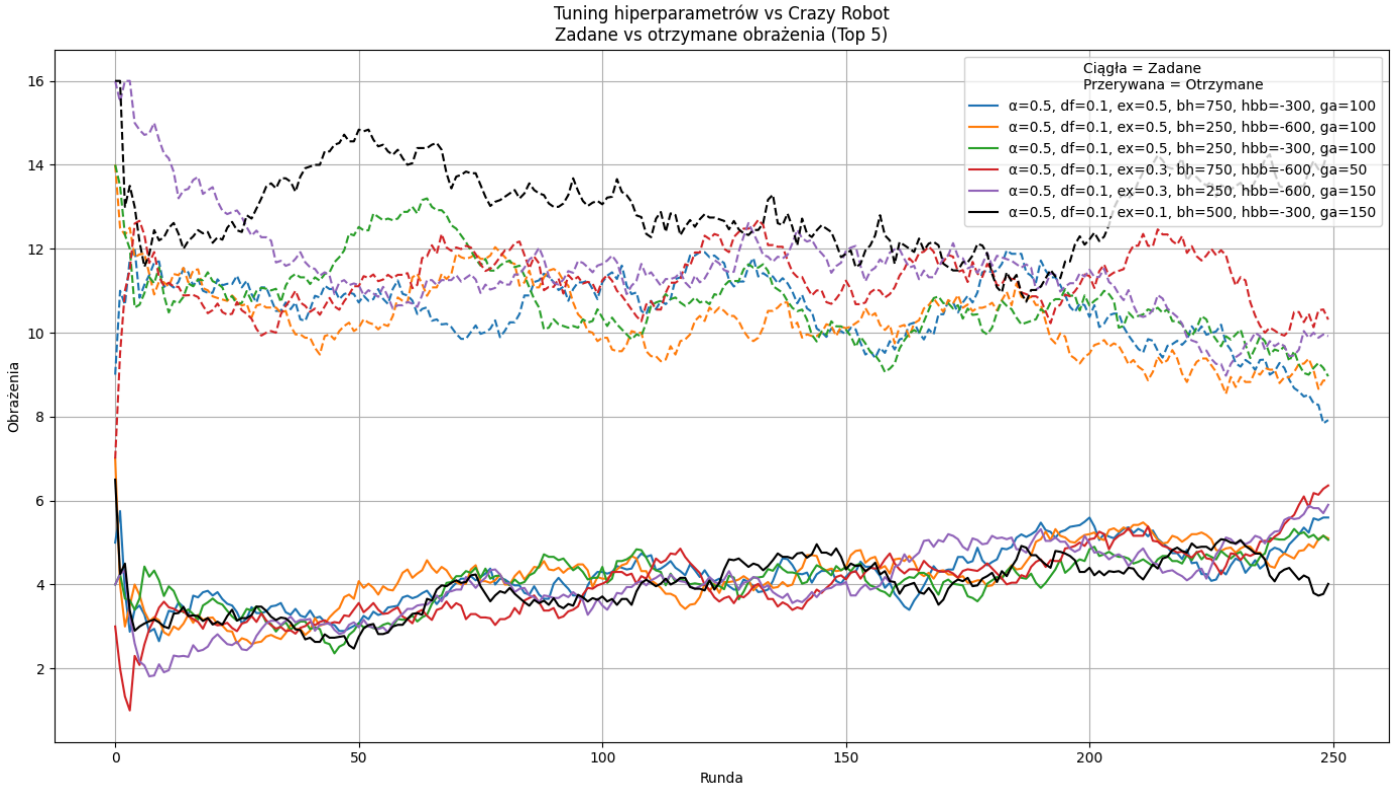
Przeszedłem do drugiego etapu optymalizacji, czyli optymalizacji nagród. Przetestowałem iloczyn kartezjański wcześniej wybranych wartości hiperparametrów zapisanych w tabeli oraz następujących wartości:

- bulletHitReward∈ {250, 500, 750} (określone na wykresach jako bh)
- hitByBulletPenalty ∈ {-300, -600} (określone na wykresach jako hbb)
- gunAlignmentReward ∈ {50, 100, 150} (określone na wykresach jako ga)

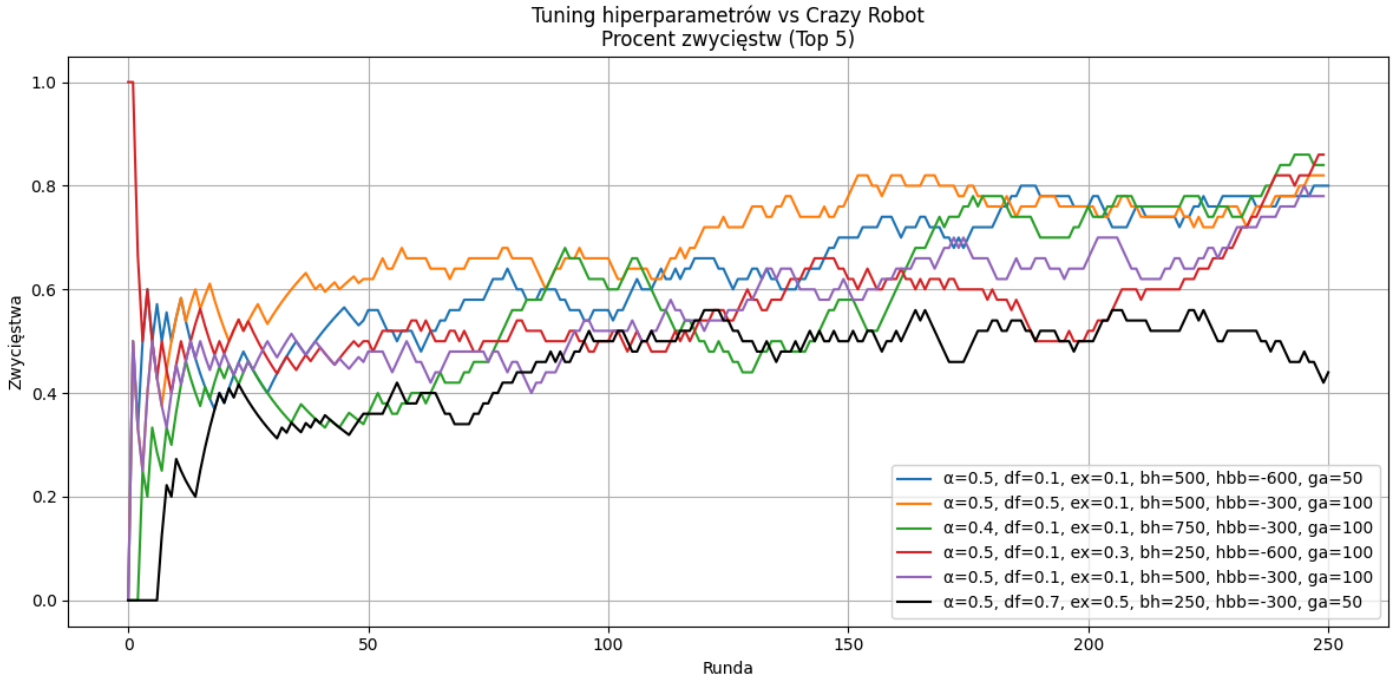
Z powodu większej liczby kombinacji do przetestowania przeprowadziłem mniejszą liczbę bitew na zestaw, po 250. Tak jak poprzednio stworzyłem wykresy:



W porównaniu z poprzednim finetuningiem widzimy dalszą redukcję w różnicy kąta działa o około 5 stopni (dla 250 rund oczywiście).



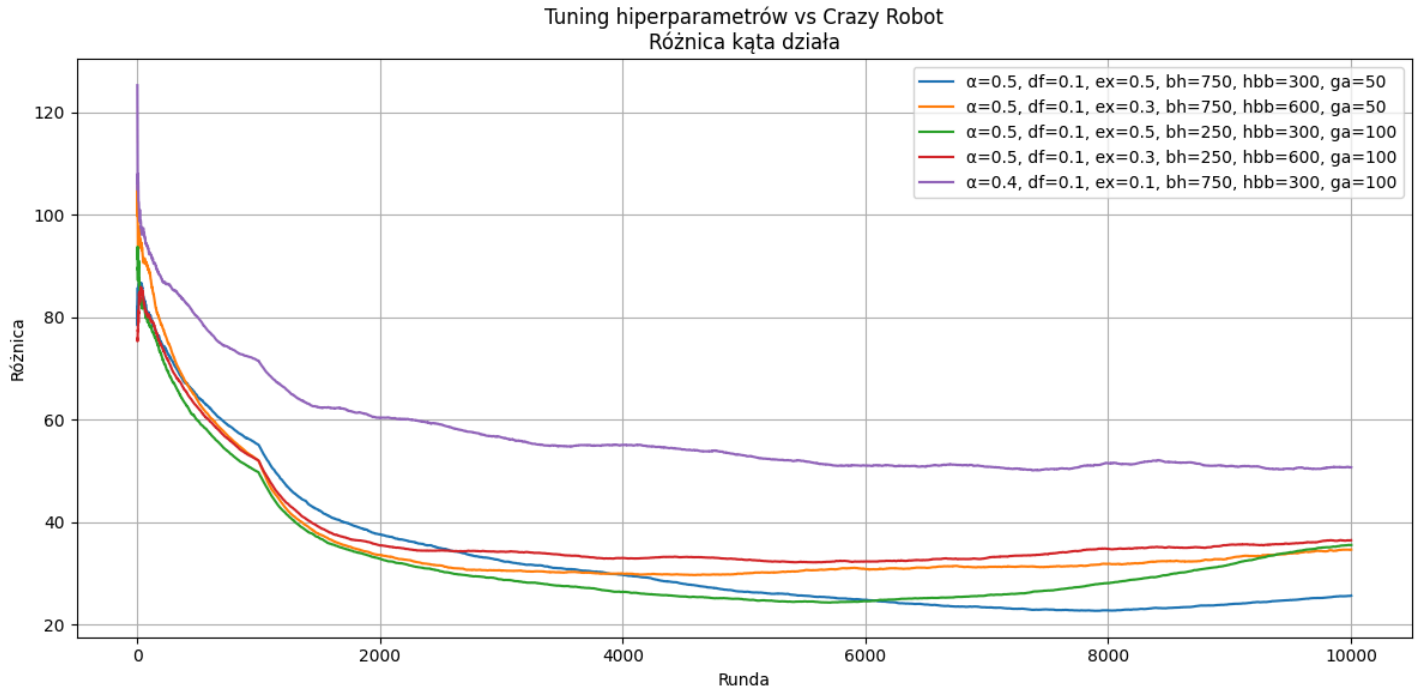
Dzięki dobranym parametrom udało się zauważalnie zredukować liczbę otrzymywanych obrażeń.



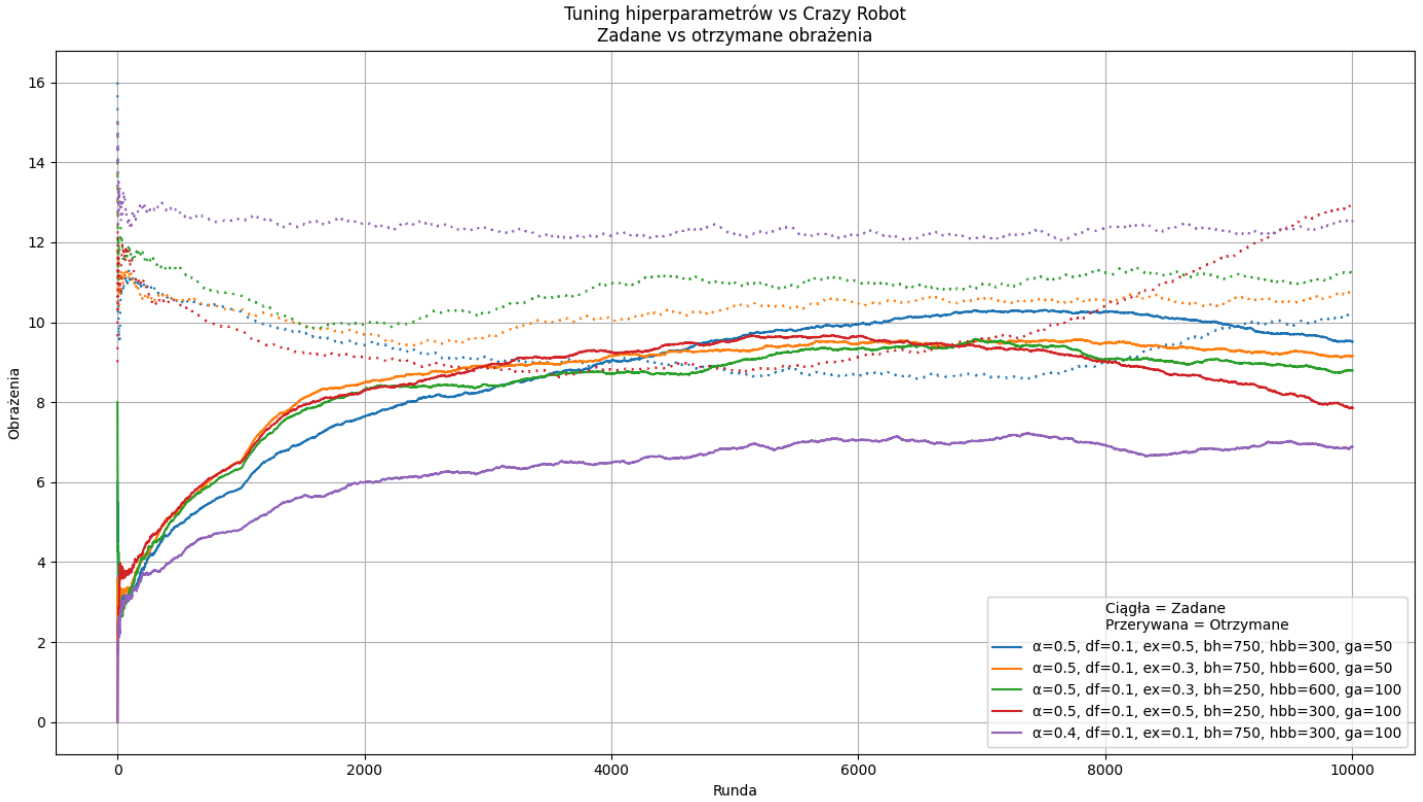
Procent zwycięstw wzrósł o kilka procent. Na podstawie tych wykresów wybrałem 5 zestawów parametrów radzących sobie najlepiej:

alpha	discountFactor	experimentRate	bulletHitReward	hitByBulletPenalty	gunAlignmentReward
0.5	0.1	0.3	750	-600	50
0.5	0.1	0.3	250	-600	100
0.5	0.1	0.5	250	-300	100
0.5	0.1	0.5	750	-300	50
0.4	0.1	0.1	750	-300	100

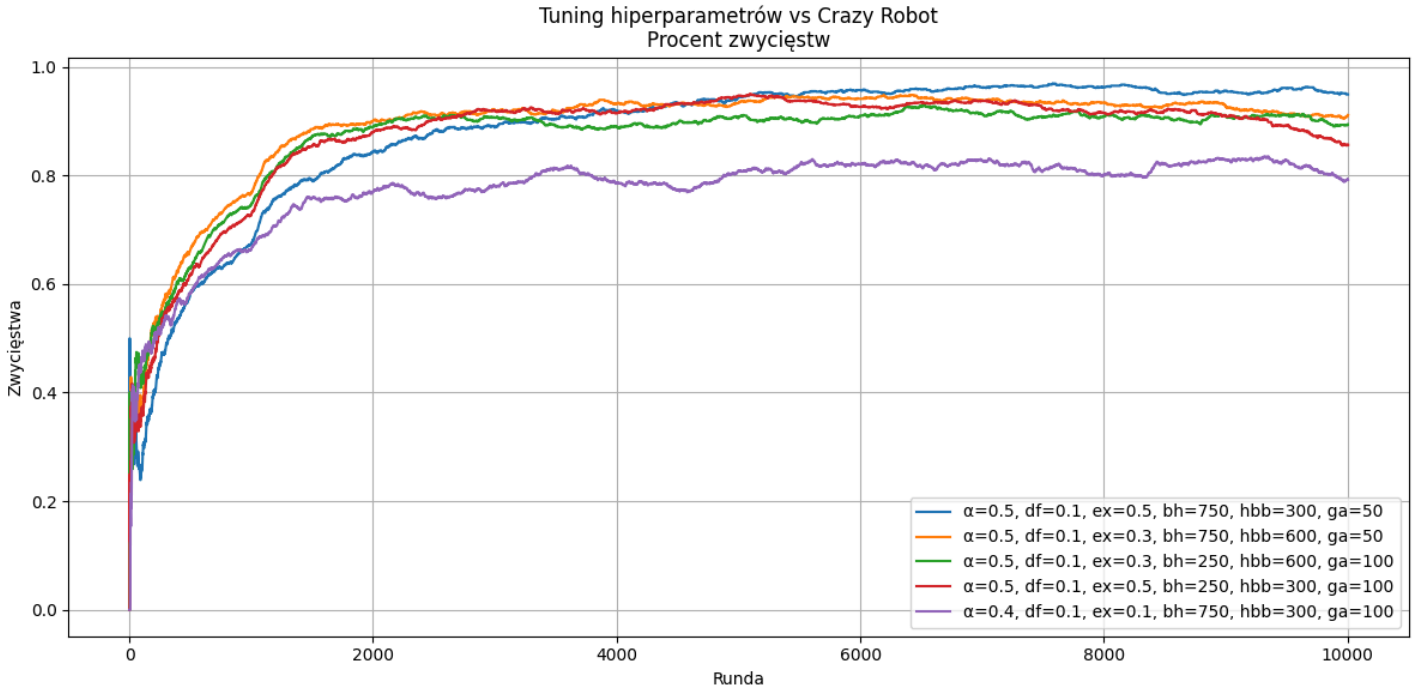
Na tych zestawach przeprowadziłem pełny trening trwający 10000 rund. Poniżej prezentuję wykresy z tego treningu:



W najlepszym przypadku osiągamy bardzo dobrą średnią różnicę kąta działa około 25 stopni.



Widzimy że dla niektórych zestawów parametrów robotowi udawało się zadawać więcej obrażeń niż otrzymywać.



Osiągamy bardzo dobry odsetek zwycięstw około 95%. Zdecydowanie najlepiej poradził sobie zestaw **alpha=0.5, df=0.1, ex=0.5, bh=750, hbb=300, ga=50** (niebieska linia) dla około 8000 rund. Ten więc zestaw wykorzystałem podczas przeprowadzania bitew z innymi robotami. Udało mu się zadawać więcej obrażeń niż przyjmować oraz wygrywać w około 95% bitew. Po 8000 rund jego wydajność zaczęła lekko słabnąć. Z tego powodu przed przeprowadzeniem bitew z innymi robotami uciąłem trening na 8000 bitew.

Wyniki

W celu przetestowania sprawności robota przeprowadziłem bitwy po 1000 rund z kilkoma typami robotów:

vs Crazy Robot - 949/1000 wygranych

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds
1st	agh.reinforced.ReinforcedLearningRobotLoa...	114366 (68%)	47350	9470	49537	7703	270	35	949	51
2nd	sample.Crazy	53505 (32%)	2550	510	44287	668	5318	171	53	947

vs Corners Robot - 352/1000 wygranych

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds
1st	sample.Corners	160138 (58%)	32400	6480	106235	14958	65	0	648	352
2nd	agh.reinforced.ReinforcedLearningRobotLoa...	116052 (42%)	17600	3520	85421	9410	101	0	352	648

vs Fire Robot - 300/1000 wygranych

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds
1st	sample.Fire	164295 (57%)	35000	7000	106096	16196	2	0	701	299
2nd	agh.reinforced.ReinforcedLearningRobotLoa...	121985 (43%)	14950	2990	95547	8432	66	0	300	700

vs Veloci Robot - 36/1000 wygranych

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds
1st	sample.VelociRobot	181190 (83%)	48200	9640	84818	16724	19572	2237	964	36
2nd	agh.reinforced.ReinforcedLearningRobotLoa...	37071 (17%)	1800	360	33779	525	606	0	36	964

vs Spin Bot - 1/1000 wygranych

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds
1st	sample.SpinBot	177739 (87%)	49950	9990	97109	19472	1159	59	999	1
2nd	agh.reinforced.ReinforcedLearningRobotLoa...	25581 (13%)	50	10	25359	27	134	0	1	999

Wnioski

- Przy pomocy uczenia ze wzmocnieniem udało się stworzyć robota który nie najgorzej radził sobie w środowisku Robocode. Dalszy finetuning hiperparametrów oraz wartości nagród zapewne pozwoliłby osiągnąć jeszcze lepsze wyniki.

- Udało się osiągnąć 95% wygranych przeciwko Crazy Robotowi, co jest doskonałym wynikiem. Niestety wygląda na to że nie generalizuje się to na inne boty.

- Prostszą wersją robota która zapewne poradziłaby sobie lepiej w bitwach jest wersja w której sterowanie korpusem czołgu jest obsługiwane jakimś prostym algorytmem, a uczenie obejmuje jedynie ruchy wieżyczki i strzały.

- Robot zapewne wykazał by się lepszą efektywnością przeciwko innym robotom jeśli na nich również zostałby przeprowadzony trening i tuning hiperparametrów.