ALHE
DOKUMENTACJA KOŃCOWA

Skład zespołu:

- *Antoni Charchuła 283713*
- *Jakub Tokarzewski 283778*

SPIS TREŚCI

1. Problem

W ramach projektu należy zaimplementować algorytm przeszukiwania (np. algorytm genetyczny/ewolucyjny), który dla danego poziomu finansowego np. 80 pkt. wybierze taką trójkę zawodników napastnik + skrzydłowi, którą charakteryzuje największa liczba strzelonych bramek.

Dane:
https://www.kaggle.com/hugomathien/soccer/home
Link do repozytorium z kodem:
https://github.com/tokerson/ALHE-Perfect-offensive

Zadanie polega na dobraniu odpowiedniej trójki zawodników w ataku, tak żeby stworzyć idealną ofensywę na danym „poziomie finansowym". Zawodnicy definiowani są jako zbiór trzech parametrów (oceny ogólnej, siły strzałów, wykończenia). Zakładamy że dwóch zawodników o tych samych parametrach jest nierozróżnialnych. Jakość ofensywy określa się poprzez liczbę strzelonych bramek lub liczbę oddanych strzałów, przez trójkę zawodników o zakładanych parametrach. Poziom finansowy definiuje maksymalną średnią wartość ocen ogólnych zawodników.

2. Analiza danych

Przed rozpoczęciem badań było konieczne przefiltrowanie bazy danych, aby posiadać jedynie mecze, które mają sensowne dane. Mianowicie, zostały odrzucone mecze zawierające NULL'e w istotnych dla nas polach np. w miejscu, któregoś z graczy ofensywnych.
Z pozostałych meczów zostały wyabstrahowane trójki graczy ofensywnych - na pozycji 9, 10 i 11 oraz ilość strzelonych przez nich bramek lub strzałów na bramkę. W trakcie analizy okazało się także, że ilość strzałów na bramkę jest często mniejsza niż liczba goli, dlatego założono, że żeby otrzymać ich prawidłową ilość to należy dodać do poprzedniej wartości ilość strzelonych bramek.

Następnie w każdym meczu przemapowano każdego zawodnika na trzy jego parametry - OVERALL_RATING, SHOT_POWER oraz FINISHING. W bazie danych istniało wiele pomiarów dla danego użytkownika, dlatego ostatecznie brano te wartości, które posiadały najwcześniejszą datę oraz nie posiadały NULLi na wcześniej wymienionych polach.

Zawodnicy nierozróżnialni są interpretowani jako ci, którzy mają takie same parametry.

Poszukiwane rozwiązanie zostało zdefiniowane jako wektor zawierający parametry trzech zawodników o największej ilości strzelonych bramek na mecz lub największej ilości strzałów na bramkę.

Istnienie danej trójki definiują nam rozegrane mecze.

Po obróbce danych ostatecznie przekazanych zostało 15921 meczów w postaci:

75,72,71,73,81,71,72,81,72,2.0
GRACZ 1 GRACZ 2 GRACZ 3 WYNIK

W projekcie wymagane także było stworzenie przestrzeni przeszukiwań. Dlatego założono, że sąsiadem meczu jest drugi, który zawiera przynajmniej jednego takiego samego zawodnika.

Aby nie musieć za każdym razem tworzyć obrobionych danych oraz struktur zawierających mecz i jego sąsiadów, użyto biblioteki Pickle, która zapamiętuje dany obiekt do pliku txt, który przy kolejnym uruchomieniu programu, można z łatwością załadować jako obiekt klasy.

3. Rozwiązanie

3.1. Przeszukiwanie liniowe

Aby uzyskać rozwiązanie w przeszukiwaniu liniowym należało przeszukać jednokrotnie listę meczów, zapamiętując dotychczas najlepszy wynik, biorąc przy tym pod uwagę narzucony maksymalny koszt danej trójki. Wynik przeszukiwania liniowego posłużył w projekcie do sprawdzania czy otrzymany wynik jest poprawny.

3.2. Symulowane wyżarzanie

Do implementacji symulowanego wyżarzania użyto biblioteki Simanneal.

https://github.com/perrygeo/simanneal

Aby z niej skorzystać należy stworzyć klasę dziedziczącą po Annealer z trzema funkcjami:

- **__init__** - konstruktor w którym inicjalizujemy stan, od którego zaczynamy poszukiwanie. Jako stan używana jest krotka zawierająca dziewięć elementów - 3 zawodników z ich parametrami. Wynik danej trójki uzyskuje się odwołując się do słownika z danymi przekazanego do klasy, który jako klucz przyjmuje krotkę, a zwraca wynik danej trójki zawodników.
- **move** - funkcja definiująca przejście z jednego stanu do drugiego. Wybierany jest dowolny sąsiad aktualnego stanu. W przypadku nawrotu ze stanu (ze względu na jego słabą jakość) ponowne losowanie klucza ominie klucz, który wylosowano poprzednio
- **energy** - funkcja definiująca jakość stanu (im mniejsza wartość tym lepiej). Jeśli koszt danego stanu jest większy niż koszt zadany w wywołaniu, wartość takiego stanu jest określana jako koszt tego stanu (czyli w okolicach 60-90 zależnie od stanu). W przeciwnym wypadku od liczby 20 (liczba dowolna byle była większa od maksymalnej wartość w bazie danych oraz dużo mniejsza od kosztów stanów)

odejmuje się wynik danej trójki. Dzięki temu im wyższy wynik posiadała dana trójka tym mniejsza będzie energia, czyli tym lepsza jakość stanu dla algorytmu.

W algorytmie dodano parametr stagnacji - czyli jeśli jakość stanu nie zwiększy swojej wartości przez tyle iteracji ile zostało przekazanych w parametrze, to algorytm kończy swoje działanie.

Dodatkowo dla każdej egzekucji algorytmu można sprecyzować ilość kroków, po których algorytm przestanie pracować. Jako krok rozumie się przejście do stanu i obliczenie dla niego energii.

Można także zmieniać temperaturę maksymalną oraz minimalną. Jednak w przypadku testów głównie był wykorzystywany automatyczny przydział temperatur, który był zawarty w bibliotece.

3.3. Algorytm ewolucyjny

Do implementacji algorytmu ewolucyjnego wykorzystana została biblioteka DEAP. Udostępnia ona szereg wbudowanych funkcji, jednak w przypadku działania na nietypowym typie danych, jak w przypadku tego projektu, wiele funkcji należy zaimplementować samemu i połączyć z funkcjami bibliotecznymi.

Zaimplementowane zostały własne rozwiązania do:
- tworzenia nowego osobnika
- krzyżowania osobników
- mutacji
- oceny

Wywołując algorytm podajemy mu informacje o parametrze stagnacji i o najlepszym osobniku dla danego zakresu cenowego. Parametr stagnacji mówi o tym po ilu generacjach w przypadku braku poprawy najlepszego osobnika należy przerwać działanie. Drugim kryterium kończącym pracę algorytmu jest znalezienie osobnika o podanej (najlepszej dla przedziału cenowego) ocenie ogólnej.

Po zakończeniu działania algorytm zwraca informacje na temat liczby generacji, które powstały podczas poszukiwań, najlepszego osobnika oraz oceny najlepszej znalezionej trójki.

3.3.1. Tworzenie nowego osobnika

Tworzenie nowego osobnika odbywa się poprzez losowanie go z naszej przestrzeni przeszukiwań. Rozważane było rozwiązanie losowania cech piłkarzy i na tej podstawie tworzenia nowej trójki (nowego osobnika). Powodowałoby to jednak częste tworzenie trójek niewystępujących w naszej bazie danych, co skutkowałoby ich słabą oceną i brakiem możliwości efektywnego ich krzyżowania czy ewolucji.

3.3.2. Krzyżowanie

Krzyżowanie osobników odbywa się na zasadzie wymiany ostatnimi zawodnikami między dwoma trójkami. Decyzja o krzyżowaniu danej pary odbywa się z pewnym prawdopodobieństwem podanym przy wywołaniu algorytmu. Krzyżowane są osobniki umieszczone na liście osobników “reprodukujących” obok siebie. Rozważana była opcja sortowania listy, aby do krzyżowania wybierać osobniki o podobnej ocenie, co potencjalnie dawałoby szansę na wygenerowanie dobrego osobnika. Mogłoby to jednak spowodować spowolnienie działania algorytmu ze względu na konieczność sortowania listy przy produkowaniu każdej nowej generacji. Biorąc pod uwagę czas wykonywania się algorytmu przeszukiwania liniowego zrezygnowano z pomysłu sortowania.

3.3.3. Mutacja

Mutacja osobnika polega na zamianie go w jednego z jego sąsiadów z przestrzeni przeszukiwań. Takie rozwiązanie jest efektywne, ze względu na to iż przestrzeń przeszukiwań zorganizowana jest jako słownik, więc dostęp do sąsiadów elementu ma złożoność O(1). Wybór dowolnego sąsiada z listy również nie jest kosztowną operacją. Taka strategia ewolucyjna daje nam pewność eksploracji przestrzeni przeszukiwań. Podobnie jak przy krzyżowaniu, każdy osobnik z listy potomków ma szansę na zmutowanie, z pewnym prawdopodobieństwem podawanym przy starcie algorytmu.

3.3.4. Ocena jakości osobnika

Algorytm ewolucyjny stara się maksymalizować wartość oceny, która w zależności od wariantu algorytmu jest liczbą oddanych celnych strzałów na bramkę, bądź liczbą strzelonych goli przez daną trójkę zawodników. Jakość danego osobnika jest jeszcze zależna od jego średniej oceny ogólnej. Jeżeli przekracza ona maksymalny koszt podany przy starcie algorytmu, osobnik dostaje ocenę 0, czyli minimalną możliwą. Ze względu na charakterystykę działania krzyżowania, mogą powstawać osobniki nie znajdujące się w naszej przestrzeni przeszukiwań. W takim przypadku ich ocena wynosi 0.

3.3.5. Selekcja

Algorytm obsługuje dwa tryby selekcji: turniejowy z możliwością dostosowania wielkości turnieju oraz wybór n najlepszych. Do nowej populacji wybierane są osobniki z populacji potomnej, poddawanej krzyżowaniu i mutacji oraz najlepszy osobnik z poprzedniej populacji. Dzięki temu przy wyborze n najlepszych nie tracimy najlepszego osobnika,a przy wyborze turniejowym ma on dużą szansę znalezienia się w kolejnej populacji.

4. Analiza skuteczności zaimplementowanych rozwiązań

4.1. Przeszukiwanie liniowe

Przeszukiwanie liniowe kończy swoje działanie we wszystkich przypadkach ze 100% skutecznością w czasie oscylującym wokół 0.004s. Złożoność obliczeniowa tego rozwiązanie to O(n), gdzie n to liczba różnych trójek piłkarzy w bazie danych.

4.2. Symulowane wyżarzanie

W badaniach pierwszy wiersz to algorytm uruchomiony w takim trybie, który zatrzymuje się w przypadku pierwszego wystąpienia najlepszego wyniku, który został uzyskany przy pomocy przeszukiwania liniowego.

Drugi wiersz mówi o jakości algorytmu uruchomionego z taką ilością kroków jaka odpowiada czasu wykonania przeszukiwania liniowego.

Kolejne siedem wierszy to algorytm uruchomiony dla różnych wartości parametru stagnacji.

Kolejne pięć - dla różnych wartości temperatury maksymalnej.

Ostatnie pięć - dla różnych wartości temperatury minimalnej.

4.2.1 Wyniki dla strzelonych goli

GOALS Max price: 100 Linear time: 0.0039403438568115234 Result: 7.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
9543.4	30.8	0.9710000000000001	-1	0.03911387920379639	[]	10
1001.0	24.3	0.29200000000000004	-1	0	[5.0, 5.0, 4.0, 5.0, 5.0, 5.0, 5.0, 4.0, 5.0, 5.0]	10
933.6	31.0	0.268	500	0	[5.0, 5.0, 5.0, 5.0, 6.0]	5
1490.4	25.8	0.20939999999999998	1000	0	[5.0, 5.0, 6.0, 5.0, 5.0]	5
3125.6	26.4	0.244	1500	0.010938167572021484	[5.0, 5.0, 5.0, 5.0]	5
4641.6	25.2	0.266	2000	0.019101381301879883	[5.0, 5.0, 6.0, 5.0]	5
6783.4	25.2	0.246	2500	0.023725430170694988	[5.0, 6.0]	5
10628.6	26.2	0.262	3000	0.06212878227233887	[5.0, 6.0, 6.0, 6.0]	5
12688.4	24.4	0.25800000000000006	3500	0.050705790519714355	[6.0]	5
10001.0	250.0	0.3	-1	0.04500349362691244	[5.0, 6.0]	5
10001.0	200.0	0.3	-1	0.04442840814590454	[5.0]	5
10001.0	150.0	0.3	-1	0.0440207322438558	[5.0, 5.0]	5
10001.0	100.0	0.3	-1	0.042713594436645505	[]	5
10001.0	50.0	0.3	-1	0.04324474334716797	[]	5
10001.0	28.2	4.0	-1	0.041276137034098305	[5.0, 5.0]	5
10001.0	27.2	3.0	-1	0.041464149951934814	[6.0]	5
10001.0	25.8	2.0	-1	0.04084652662277222	[5.0]	5
10001.0	27.4	1.0	-1	0.04136085510253906	[5.0]	5
10001.0	26.0	0.5	-1	0.03953735033671061	[5.0, 6.0]	5

GOALS Max price: 90 Linear time: 0.0036461353302001953 Result: 7.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
8706.1	24.8	0.642	-1	0.0350755214691162	[]	10
1001.0	24.4	0.23900000000000002	-1	0	[5.0, 4.0, 5.0, 6.0, 5.0, 5.0, 4.0, 6.0, 4.0, 5.0]	10
1484.6	27.0	0.20340000000000003	500	0	[5.0, 5.0, 5.0, 5.0, 5.0]	5
1866.4	23.6	0.20800000000000002	1000	0.010350227355957031	[5.0, 5.0, 5.0, 5.0]	5
4121.2	27.6	0.238	1500	0.012282490730285645	[5.0, 5.0, 5.0]	5
6665.0	25.2	0.24	2000	0.029750943183898926	[5.0, 5.0, 6.0]	5
4067.4	25.4	0.21400000000000002	2500	0.01934492588043213	[5.0, 5.0, 5.0]	5
8432.8	26.0	0.258	3000	0.04823486010233561	[5.0, 5.0]	5
9716.2	28.2	0.22400000000000003	3500	0.0467317104339596	[6.0, 6.0]	5
10001.0	250.0	0.3	-1	0.04516512155532837	[5.0]	5
10001.0	200.0	0.3	-1	0.04438338279724121	[]	5
10001.0	150.0	0.3	-1	0.04406628608703613	[]	5
10001.0	100.0	0.3	-1	0.04426300525665283	[6.0]	5
10001.0	50.0	0.3	-1	0.04452832539876302	[5.0, 5.0]	5
10001.0	26.4	4.0	-1	0.041610431671142575	[]	5
10001.0	23.6	3.0	-1	0.040576934814453125	[5.0, 5.0]	5
10001.0	23.0	2.0	-1	0.04050978024800619	[5.0, 5.0]	5
10001.0	26.0	1.0	-1	0.04105740785598755	[5.0]	5
10001.0	25.2	0.5	-1	0.04046928882598877	[5.0]	5

GOALS Max price: 80 Linear time: 0.0036551952362060547 Result: 5.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
993.5	186.0	1.52	-1	0.004071640968322754	[]	10
1001.0	192.0	0.262	-1	0.00402379035949707	[4.0, 4.0]	10
1323.0	198.0	0.268	500	0.004292726516723633	[4.0, 4.0]	5
2082.0	176.0	0.17200000000000001	1000	0.00965970754623413	[4.0]	5
3679.2	224.0	0.19	1500	0.015451192855834961	[4.0]	5
4833.0	194.0	0.246	2000	0.019529104232788086	[]	5
6167.6	158.0	0.28600000000000003	2500	0.025053644180297853	[]	5
13042.4	194.0	0.26799999999999996	3000	0.053582382202148435	[]	5
16660.6	180.0	0.23400000000000004	3500	0.06721677780151367	[]	5
10001.0	250.0	0.3	-1	0.04435639381408692	[]	5
10001.0	200.0	0.3	-1	0.04513826370239258	[]	5
10001.0	150.0	0.3	-1	0.04529194831848145	[]	5
10001.0	100.0	0.3	-1	0.0441077709197998	[]	5
10001.0	50.0	0.3	-1	0.04548015594482422	[]	5
10001.0	162.0	4.0	-1	0.04068388938903809	[]	5
10001.0	226.0	3.0	-1	0.04228043556213379	[]	5
10001.0	222.0	2.0	-1	0.04140682220458984	[]	5
10001.0	172.0	1.0	-1	0.04148058891296387	[]	5
10001.0	194.0	0.5	-1	0.040181207656860354	[]	5

GOALS Max price: 70 Linear time: 0.004704475402832031 Result: 5.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
724.0	288.0	1.373	-1	0.0030006647109985353	[]	10
1001.0	272.0	0.20600000000000004	-1	0.0037937164306640625	[3.0, 3.0, 3.0, 2.0, 4.0, 3.0, 3.0, 3.0, 3.0]	10
1186.6	280.0	0.258	500	0	[3.0, 3.0, 3.0, 3.0, 3.0]	5
1695.0	260.0	0.20800000000000002	1000	0	[3.0, 3.0, 3.0, 3.0, 3.0]	5
4447.6	258.0	0.164	1500	0.0359800701446533 2	[3.0, 3.0, 3.0, 3.0]	5
10250.2	256.0	0.22800000000000004	2000	0.04952096939086914	[3.0, 4.0, 4.0, 4.0]	5
13398.2	248.0	0.23600000000000004	2500	0.05664889017740885	[3.0, 4.0]	5
8254.8	258.0	0.296	3000	0.034407520294189455	[]	5
13414.0	234.0	0.186	3500	0.054179747899373375	[4.0, 4.0]	5
10001.0	250.0	0.3	-1	0.04368464152018229	[3.0, 3.0]	5
10001.0	200.0	0.3	-1	0.04243389765421549	[4.0, 3.0]	5
10001.0	150.0	0.3	-1	0.042736148834228514	[]	5
10001.0	100.0	0.3	-1	0.04189443588256836	[3.0, 4.0, 3.0, 3.0]	5
10001.0	50.0	0.3	-1	0.0409540176391601 6	[]	5
10001.0	268.0	4.0	-1	0.04653081893920898	[]	5
10001.0	258.0	3.0	-1	0.040036141872406006	[3.0, 4.0]	5
10001.0	262.0	2.0	-1	0.039371093114217125	[3.0, 3.0]	5
10001.0	274.0	1.0	-1	0.0398678183555603	[4.0]	5
10001.0	254.0	0.5	-1	0.03896911938985189	[]	5

Na powyższych danych widać, że otrzymanie dobrej odpowiedzi dla czasu 0.04 powiodło się tylko dla ceny 80, gdzie tylko dwa wyniki były błędne. W przypadku ceny 70, tylko jeden wynik był poprawny, a dla reszty żaden. Jak widać jest na tyle dużo przejść ze stanów, że osiągnięcie poprawnego wyniku tylko w 1000 korkach jest bardzo trudne.

Kontynuując analizę, można zauważyć, że im wyższy parametr stagnacji tym lepsza jakość wyników. Skuteczność na poziomie 80% można otrzymać już na poziomie stagnacji równej

2500, co odpowiada zazwyczaj ~ 7000 kroków. Dla ceny maksymalnej tych kroków należy wykonać znacznie więcej ze względu na dużą ilość stanów o słabej jakości.

W przypadku temperatury maksymalnej, duże wartości sprawiają, że przez bardzo długi czas następuje okres eksploracji, co utrudnia później znalezienie maksimum lokalnego.
Zaś spadająca temperatura minimalna spowodowała wzrost jakości algorytmu, co było oczekiwane.

Niektóre rozbieżności np. wynik algorytmu dla ceny 70 i Tmax 100, były prawdopodobnie spowodowane niefortunnością wylosowania klucza startowego, dla którego stan był oddalony od maksimum globalnego lub przez losowanie sąsiadów, które wcale nie zbliżało do wyniku.

Bardzo dobry wynik dla ceny 80 prawdopodobnie był spowodowany łatwą dostępnością wyniku 5.0 w przestrzeni przeszukiwań.

4.2.2 Wyniki dla strzałów na bramkę

SHOTONS Max price: 100 Linear time: 0.0036537647247314453 Result: 13.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
1475.7	64.1	1.068	-1	0.006219482421875	[]	10
1001.0	65.2	0.29900000000000004	-1	0.00402216116587321	[11.0, 12.0, 12.0, 12.0]	10
909.8	67.8	0.404	500	0.0032148361206054688	[11.0, 11.0, 10.0]	5
2386.4	59.4	0.47000000000000003	1000	0	[12.0, 12.0, 11.0, 12.0, 12.0]	5
3947.0	64.0	0.328	1500	0.020124276479085285	[12.0, 12.0]	5
7405.8	64.0	0.264	2000	0.030267333984375	[]	5
6618.6	63.0	0.368	2500	0.027481269836425782	[]	5
5355.8	62.0	0.246	3000	0.0218808650970459	[]	5
12005.6	61.0	0.37	3500	0.04927902221679688	[]	5
10001.0	250.0	0.3	-1	0.04622087478637695	[]	5
10001.0	200.0	0.3	-1	0.045321321487426756	[]	5
10001.0	150.0	0.3	-1	0.043901491165161136	[]	5
10001.0	100.0	0.3	-1	0.043381357192993165	[]	5
10001.0	50.0	0.3	-1	0.04286961555480957	[]	5
10001.0	66.6	4.0	-1	0.041578817367553714	[]	5
10001.0	62.8	3.0	-1	0.04190373420715332	[]	5
10001.0	57.8	2.0	-1	0.04219145774841308	[]	5
10001.0	66.4	1.0	-1	0.04164919853210449	[]	5
10001.0	51.2	0.5	-1	0.0388674259185791	[]	5

SHOTONS Max price: 90 Linear time: 0.0036804676055908203 Result: 13.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
1816.8	60.9	1.241	-1	0.007519030570983886	[]	10
1001.0	61.0	0.32100000000000006	-1	0.004104900360107422	[12.0, 12.0, 10.0, 11.0, 12.0]	10
1018.6	62.8	0.23199999999999998	500	0.004810333251953125	[12.0, 11.0]	5
2414.4	56.8	0.35	1000	0.011716604232788086	[12.0, 12.0]	5
3706.8	68.2	0.31999999999999995	1500	0.015503215789794921	[]	5
5555.0	68.8	0.332	2000	0.02299365997314453	[]	5
5483.4	67.8	0.312	2500	0.02485334873199463	[12.0]	5
11059.6	67.2	0.27999999999999997	3000	0.04546699523925781	[]	5
10503.6	63.6	0.28200000000000003	3500	0.04448366165161133	[]	5
10001.0	250.0	0.3	-1	0.043302059173583984	[]	5
10001.0	200.0	0.3	-1	0.04429011344909668	[]	5
10001.0	150.0	0.3	-1	0.04967379570007324	[]	5
10001.0	100.0	0.3	-1	0.044936609268188474	[]	5
10001.0	50.0	0.3	-1	0.04898476600646973	[]	5
10001.0	70.6	4.0	-1	0.04268074035644531	[]	5
10001.0	52.4	3.0	-1	0.04462532997131348	[]	5
10001.0	66.4	2.0	-1	0.04084491729736328	[]	5
10001.0	56.0	1.0	-1	0.04022817611694336	[]	5
10001.0	58.2	0.5	-1	0.03916292190551758	[]	5

SHOTONS Max price: 80 Linear time: 0.004830360412597656 Result: 13.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
1812.3	219.0	1.214	-1	0.0078595876693725 59	[]	10
1001.0	248.0	0.28200000000000003	-1	0.0040683746337890625	[10.0, 11.0, 11.0, 12.0, 10.0, 10.0, 10.0]	10
814.2	258.0	0.278	500	0.0022182464599609375	[10.0, 12.0, 11.0, 11.0]	5
1854.2	218.0	0.268	1000	0.006919384002685547	[10.0, 12.0, 12.0, 12.0]	5
2727.0	204.0	0.35200000000000004	1500	0.011250591278076172	[]	5
8481.8	218.0	0.29400000000000004	2000	0.031587958335876465	[12.0]	5
5498.0	252.0	0.41400000000000003	2500	0.022555649280548096	[12.0]	5
7620.8	240.0	0.30999999999999994	3000	0.03275742530822754	[]	5
8375.6	212.0	0.324	3500	0.0452876091003418	[12.0]	5
10001.0	250.0	0.3	-1	0.04588971138000488	[]	5
10001.0	200.0	0.3	-1	0.05074653625488281	[]	5
10001.0	150.0	0.3	-1	0.05189957618713379	[]	5
10001.0	100.0	0.3	-1	0.05050497055053711	[]	5
10001.0	50.0	0.3	-1	0.053615903854370116	[]	5
10001.0	224.0	4.0	-1	0.044752311706542966	[]	5
10001.0	212.0	3.0	-1	0.04891438484191894	[]	5
10001.0	212.0	2.0	-1	0.040674686431884766	[]	5
10001.0	228.0	1.0	-1	0.03961724042892456	[12.0]	5
10001.0	246.0	0.5	-1	0.04306902885437012	[]	5

SHOTONS Max price: 70 Linear time: 0.003815889358520508 Result: 9.0

Steps	Tmax	Tmin	Stagnation	Avg time	Wrong results	Iterations
235.9	277.0	2.37	-1	0.000995802879333496	[]	10
1001.0	264.0	0.29900000000000004	-1	0	[8.0, 8.0, 7.0, 8.0, 8.0, 8.0, 8.5, 8.5, 8.5, 8.0, 6.0]	10
886.8	286.0	0.27199999999999996	500	0.0024721622467041016	[7.0, 6.0, 5.0, 7.0]	5
2043.2	266.0	0.3766	1000	0.0070031483968098955	[8.0, 8.0]	5
3362.2	260.0	0.34199999999999997	1500	0.018405437469482422	[8.5, 8.0, 8.0, 8.0]	5
3974.2	296.0	0.308	2000	0.019484837849934895	[8.5, 8.0]	5
5442.0	286.0	0.268	2500	0.012511253356933594	[8.0, 8.0, 8.0, 8.0]	5
6480.6	232.0	0.354	3000	0.03535866737365723	[8.0, 8.0, 8.5, 8.5]	5
7998.0	294.0	0.29000000000000004	3500	0.032611417770385745	[]	5
10001.0	250.0	0.3	-1	0.0431167125701904 3	[]	5
10001.0	200.0	0.3	-1	0.04255843162536621	[]	5
10001.0	150.0	0.3	-1	0.04243602752685547	[]	5
10001.0	100.0	0.3	-1	0.04232382774353027	[]	5
10001.0	50.0	0.3	-1	0.0411717414855957	[]	5
10001.0	294.0	4.0	-1	0.0393435001373291	[]	5
10001.0	290.0	3.0	-1	0.039641952514648436	[]	5
10001.0	252.0	2.0	-1	0.04204306602478027	[]	5
10001.0	292.0	1.0	-1	0.04069894552230835	[8.0]	5
10001.0	224.0	0.5	-1	0.037761259078979495	[]	5

Jak widać w przypadku strzałów na bramkę algorytm poradził sobie znacznie lepiej. Powodem jest na pewno większa różnorodność danych. W przypadku goli jest bardzo dużo trójek zawodników z wartością strzelonych bramek równą 0, przez co algorytm często traci kroki na porównania z takimi stanami i ciężko mu przejść do tych lepszych.

Na co warto zwrócić uwagę to fakt, że w przypadku kosztu 100 i 90 udało się uzyskać skuteczność na poziomie 50 % dla czasu zbliżonego do liniowego, co wydaje się być zadowalającym wynikiem.

4.3. Algorytm ewolucyjny

Algorytm ewolucyjny został przetestowany dla dwóch różnych selekcji i na dwóch innych zbiorach danych (jednym z liczbą goli strzelonych przez trójką i drugim z liczbą oddanych celnych strzałów). Łącznie daje to 4 różne kombinacje tych parametrów do przetestowania.
Podczas testowania zostały również prowadzone obserwacje wpływu każdego z czynników: wielkość populacji, długość stagnacji, prawdopodobieństwo krzyżowania, prawdopodobieństwo mutacji.

Tabele przedstawione w kolejnych sekcjach tego nagłówka będą posiadały następujące kolumny:

- Population size - wielkość populacji;
- Stagnation - ustawiona długość stagnacji;
- Avg time - średni czas wykonania algorytmu w *k* próbach (*k* zdefiniowane w kolumnie *Repetitions*);

- Avg generations - średnia liczba generacji, które zostały stworzone podczas trwania algorytmu w *k* próbach;
- Avg Score - średni wynik najlepszego osobnika uzyskanego w *k* próbach;
- Highest Score - wynik najlepszego osobnika uzyskanego w *k* próbach;
- Price - cena (średnia ocena ogólna) najlepszego osobnika uzyskanego w *k* próbach;
- Repetitions - liczba wywołań algorytmu dla tych samych parametrów wywołania;
- Accuracy - stosunek liczby znalezionych najlepszych rozwiązań do liczby nieznalezionych, w *k* próbach.

Ze względu na liczność wyników testowych nie zostały one umieszczone w tej dokumentacji, aby nie zaburzać jej czytelności. Wyniki znajdują się w pliku evo_results.txt. Każda z sekcji tabel oznaczona jest nagłówkiem informującym o tym z jakimi parametrami był testowany algorytm ewolucyjny. W treści dokumentacji znajduje się jedynie omówienie rezultatów i ich analiza.

Legenda oznaczeń:
- Selection - selekcja, może byc “n BEST” lub “TOURNAMENT”
- Data - dane, na których chcemy pracować (“GOALS”, “SHOTONS”) w zależności czy szukamy trójki która strzeliła najwięcej goli, czy tej która oddała najwięcej celnych strzałów.
- CRSPB - prawdopodobieństwo krzyżowania
- MTNPB - prawdopodobieństwo mutacji
- MAX-PRICE - górne ograniczenie co do ceny trójki
- TOURNAMENT_SIZE - wielkość turnieju (tylko przy selekcji turniejowej)

Dla każdej sekcji wygenerowane zostało 6 tabel pokazujących wpływ różnych parametrów na uzyskiwane wyniki. Pierwsza i druga tabela przedstawiają wyniki dla zmiennej wielkości populacji. Trzecia i czwarta tabela prezentuje wpływ wydłużenia stagnacji na jakość wyników. Piąta tabela ukazuje znaczenie prawdopodobieństwa krzyżowania, a szósta prawdopodobieństwa mutacji. Jeżeli dwie ostatnie tabele są nieczytelne, pomocne może się okazać wyłączenie opcji “soft wrapping” w edytorze tekstu.

Obliczenia dla jednej konfiguracji parametrów obliczane są wielokrotnie i brane pod uwagę ich są średnie wyniki, aby zminimalizować wpływ nieszczęśliwie losowych przebiegów.

Dla przypomnienia dodam, że algorytm kończy liczenie w przypadku przekroczenia czasu stagnacji, bądź po znalezieniu szukanego elementu.

4.3.1. Selekcja n najlepszych, celne strzały

Wielkość populacji
Analizując wpływ wzrostu liczebności populacji można zauważyć, że zwiększanie populacji ma wymierne korzyści, dla mniejszych wartości MAX-PRICE (przedział 80-70). Jeżeli ten parametr jest wysoki możemy zauważyć nieznaczną (około 15 punktów procentowych) poprawę celności przy zwiększaniu liczebności populacji. Poprawa przestaje być widoczna

w okolicach 70-100 osobników. Poprawa celności spowodowana jest tym, że więcej osobników eksploruje (poprzez mutację) przestrzeń przeszukiwań dla tego samego parametru stagnacji. Naturalnie powoduje to przeanalizowanie większej ilości możliwych trójek.

Długość stagnacji

Bardzo dobrze widoczny jest wpływ długości stagnacji na celność rozwiązań. Przy zwiększaniu parametru widoczna jest wysoka poprawa średnich uzyskiwanych wyników. Ciekawej obserwacji można dokonać dla limitu MAX-PRICE=70. Widać tam, że wyniki z przedziału wartości stagnacji 400-460 są lepsze od tych z przedziału 0-400. Co więcej, można dokonać kolejnej obserwacji mówiącej, że średnia liczba generacji powstałych podczas obliczeń jest większa, bądź mniejsza lecz zbliżona, od ustawionego limitu stagnacji. Oznacza to, że większość pomiarów trwała tak długo jak mogła i kończyła się niepowodzeniem. Z tej obserwacji można przypuszczać, że dalsze zwiększanie długości stagnacji dla limitu MAX-PRICE 70 wykazałoby polepszenie uzyskiwanych wyników.
Z tego wynika, że dla tego zbioru danych przy niskim ograniczeniu cenowym (MAX-PRICE niskie) warto ustawić dłuższą długość możliwej stagnacji, aby uzyskać poprawny wynik, gdyż to on odgrywa w tym przypadku znaczącą rolę.

Prawdopodobieństwo krzyżowania

Z tabel wynika, że zmiana prawdopodobieństwa krzyżowania ma znikomy wpływ na jakość otrzymywanych rozwiązań. Może nas to informować o tym, że sposób krzyżowania zastosowany w tym algorytmie ewolucyjnym nie jest optymalny (szczegóły implementacyjnie w pkt. 3.3.2.). Z tej obserwacji może wynikać, że krzyżowanie często doprowadza do powstawania osobników niebędących elementami przestrzeni przeszukiwań. Takie elementy niestety spowalniają rozwój populacji, bo aby były użyteczne należy je wyeliminować z populacji, albo skrzyżować z innym elementem. Zmniejszanie prawdopodobieństwa krzyżowania zmniejsza średni czas trwania algorytmu. Spowodowane jest to przypuszczalnie tym, że mniej krzyżowań musi się wykonać podczas generowania nowego pokolenia. Operacja krzyżowania nie jest kosztowna, ale zmniejszenie liczby tych operacji o połowę korzystnie wpływa na średni czas trwania algorytmu.

Prawdopodobieństwo mutacji

Ten czynnik ma duży wpływ na jakość otrzymywanych rozwiązań, szczególnie dla wysokiej wartości ograniczenia MAX_PRICE. W przypadku Selekcji n najlepszych mamy zapewnione włączenie do następnej populacji najlepszego osobnika z poprzedniej populacji, bądź populacji potomnej. Oznacza to, że nawet jeśli zostanie on zmutowany to jego oryginalna kopia trafi do następnej populacji. Zwiększenie prawdopodobieństwa mutacji zwiększa ilość przeanalizowanych osobników. Przy prawdopodobieństwie mutacji równym 1 ten algorytm ewolucyjny przypomina działaniem algorytm błądzenia przypadkowego z wielu miejsc jednocześnie. Zmniejszanie prawdopodobieństwa mutacji zwiększa czas trwania algorytmu, ponieważ więcej generacji musi wytworzyć aby znaleźć rozwiązanie.

4.3.2. Selekcja turniejowa, celne strzały

Tabele pokazujące wpływ wielkości turnieju na jakość rozwiązań wydatnie pokazują, że najoptymalniejszą wielkością turnieju jest 2. Dla tej wartości średnia otrzymywanych wyników plasuje się w okolicach 12.5 w 20 powtórzeniach. Szukana wartość przy tych obliczeniach to 13.0. Przy zwiększaniu liczebności turnieju można zauważyć spadek średniej jakości rozwiązań poniżej 12.0. Zmiana liczebności turnieju nie wydaje się powodować zmiany czasu trwania algorytmu. W wynikach występuje jedna anomalia, dla wielkości turnieju 2 i wielkości populacji 10. Widać, że średnia liczba generacji w tamtym przypadku to 1557, podczas gdy stagnacja była ustawiona na 300. Oznacza to, częste poprawy najlepszego osobnika, co powoduje resetowanie licznika stagnacji, czyli przedłużenie trwania algorytmu. Średnia otrzymanych wyników plasuje się niestety bardzo nisko, bo wynosi tylko 2.4, z czego wynika, że poprawy jakości rozwiązań były nieznaczne. Można z tego wnioskować że dla wielu wywołań z tymi parametrami algorytm startował w otoczeniu słabych rozwiązań, w których optimach lokalnych utykał. Jest to możliwy scenariusz ze względu na bardzo małą liczebność populacji wynoszącą 10.

Wielkość populacji

Obliczenia pokazują że zwiększanie wielkość populacji ma nieznaczny wpływ na poprawę wyników tego wariantu algorytmu. Przy selekcji turniejowej do populacji potomków i kolejnej populacji często będą trafiały dobrze ocenione osobniki, często te same w kilku egzemplarzach. W takim wypadku algorytm zamiast eksplorować przestrzeń przeszukiwań ma tendencję do jej eksploatowania, przez co dla tych samych parametrów daje lepsze rezultaty niż selekcja n najlepszych. Potwierdzeniem teorii o eksploatowaniu jest wysoka wartość średniej liczby generacji, co oznacza, że często następowała poprawa najlepszego osobnika.

Długość stagnacji

Podobnie jak przy strategii selekcji n najlepszych tak i w tym przypadku długość stagnacji ma duży wpływ na jakość rozwiązań. Im dłuższa stagnacja tym dokładniejsze otrzymywane wyniki. Wynika z tego, że jeżeli zależy nam na jak najbardziej dokładnym wyniku kosztem czasu, należy wydłużyć stagnację.

Prawdopodobieństwo krzyżowania

W przeciwieństwie do strategii selekcji n najlepszych, w tym przypadku spadek prawdopodobieństwa krzyżowania wydatnie zmniejsza dokładność otrzymywanych rozwiązań. Może to wynikać z tego, że przy tej strategii w populacji jest często wiele dobrych osobników, które po skrzyżowaniu ze sobą mogą dawać nam kolejnego dobrego osobnika. Losowy wybór sąsiada z przestrzeni przeszukiwań może dawać nam mniejszą pewność na poprawę.

Prawdopodobieństwo mutacji

Wyniki jawnie pokazują, że prawdopodobieństwo mutacji ma znikomy wpływ na jakość otrzymywanych rozwiązań. Może być to spowodowane tym, że podczas mutacji często jest

generowany gorszy osobnik od poprzedniego i w ten sposób wygenerowany potomek przegrywa przez to w turnieju i nie jest brany do kolejnej populacji.

4.3.3. Gole

Przestrzeń przeszukiwań skupiająca się na golach zamiast celnych strzałach zawiera o wiele więcej osobników z wynikiem 0. Dla tej przestrzeni algorytm selekcji n najlepszych radzi sobie bardzo źle, dla tych samych parametrów jak w przestrzeni ze strzałami celnymi. Spowodowane jest to równomiernym eksplorowaniem przestrzeni z *n* różnych miejsc. O wiele lepiej spisuje się w tej przestrzeni ponownie selekcja turniejowa, która będzie skupiała się na eksploatowaniu miejsca wychodzącego od osobników często wygrywających turniej, czyli tych z lepszą oceną. Będzie to powodowało częste analizowanie w jednej generacji wielu sąsiadów punktów wysoko ocenionych.

Wpływ pozostałych parametrów na działania algorytmów jest podobny do wpływu przedstawionego w analizach 4.3.2. oraz 4.3.3.

4.3.4. Podsumowanie

Przestrzeń przeszukiwań trójek piłkarzy ma taką cechę, że sąsiadami dobrych osobników są często inni wysoko ocenieni osobnicy, ponieważ różnią się tylko jednym piłkarzem. Ta cecha powoduje, że jest mało optimów lokalnych, a więc nie powinno nam zależeć tak bardzo na eksploracji przestrzeni jak na eksploatacji miejsc, które potencjalnie mają szansę prowadzić do najlepszych trójek. Nie powinno nikogo dziwić zatem, że selekcja turniejowa odniosła lepsze rezultaty pod względem jakości otrzymanych wyników od selekcji n najlepszych, dla tych samych parametrów. Stało się to jednak kosztem czasu trwania algorytmu, który był większy ze względu na eksploatację, czyli częste znajdowanie lepszego osobnika i zerowanie licznika stagnacji. Kolejny wniosek, który się nasuwa po przeprowadzeniu powyższej analizy jest taki, że chcąc optymalizować działanie algorytmu szukającego najlepszego rozwiązania (nie znając go wcześniej) należy skupić się na doborze odpowiedniej długości stagnacji, ponieważ ona ma największy wpływ na poprawę wyników. Należy pamiętać, że zbyt długa stagnacja znacznie wydłuży działanie algorytmu. Drugim co do ważności parametrem do optymalizacji jest wielkość populacji.

5. Porównanie działania zaimplementowanych algorytmów

Jako wyniki brano najskuteczniejszy wynik uzyskany dla danego kosztu.
Czas działania przeszukiwania liniowego ~ 0.004 s

GOALS	Symulowane wyżarzanie	Algorytm ewolucyjny (selekcja turniejowa)	Algorytm ewolucyjny (selekcja n najlepszych)
Koszt 100	0.0507s / 90 %	0.9472s / 100%	0.1833s / 50%
Koszt 90	0.0467s / 80%	0.5414s / 100%	0.2539s / 40%
Koszt 80	0.0195s / 100%	0.3174s / 100%	0.1004s / 90%
Koszt 70	0.0344s / 100%	-	0.2251s / 20%
SHOTONS			
Koszt 100	0.0302s / 100%	0.1785s / 100%	0.0773s / 95%
Koszt 90	0.0155s / 100%	0.1679s / 100%	0.0863s / 95%
Koszt 80	0.0112s / 100%	0.3338s / 100%	0.1694s / 80%
Koszt 70	0.0326s / 100%	-	0.2069s / 40%

6. Dodatkowe wnioski

Jak widać, nie udało się w projekcie zaimplementować takich algorytmów, które uzyskałyby tak dobre rezultaty jak w przeszukiwaniu liniowym. Chcąc zbliżyć się do podobnego czasu wykonania, jedynie w przypadku symulowanego wyżarzania się to udało, ale i tak trzeba liczyć się z dużą fluktuacją jakości wyników. Powodem może być niekorzystna baza danych, która ma wiele wyników słabych oraz małą ilość tych dobrych. Dodatkowym narzutem jest konieczność wykonywania wielu porównań podczas wykonywania obu algorytmów, co jest bardziej kosztowne, niż n porównań w przeszukiwaniu liniowym przy tak małym problemie.
Algorytmy radzą sobie znacznie gorzej dla niższych ograniczeń cenowych. W przypadku symulowanego wyżarzania jest to spowodowane dużym spadkiem jakości stanów, które przewyższają zadaną cenę. Skutkuje to tym, że algorytmowi ciężko przemieszczać się w poszukiwaniu rozwiązań optymalnych.
Patrząc także na zbiory danych, od razu widać, że rozwiązania dla strzałów na bramkę są znacznie lepsze. Jest to spowodowane większą różnorodnością wyników. W danych dot. strzelonych bramek, bardzo dużo rekordów ma sumę bramek równą 0, co implikuje stagnacje w poszukiwaniach.