Zasady programowania strukturalnego II – projektowanie

Wykonał
Denys Shushpanov
Grupa MT-121 rok. ak. 16/17

Prowadzący
dr. inż Adam Piechna

Temat projektu: Algorytm genetyczny

Nazwa projektu: «Cyber farma»

Opis pojęć dotyczących projektu

Dzieci - osobniki posiadające część genomu od pary z najlepszych botów poprzedniego pokolenia. Jest ich 18 od każdej pary «rodziców».

Rodzice - najlepsze osobniki z poprzedniego pokolenia, które biorą udział w skrzyżowaniu.

Genom - lista ze 100 liczb od 0 do 63, indywidualna dla każdego osobnika, które odpowiadają za ściśle określone zachowanie osobnika.

Cykl - okres życia jednego pokolenia.

Pole - tablica dynamiczna dwuwymiarowa, reprezentująca obszar, po którym mogą się przemieszczać osobniki i na którym są rozmieszczone klatki z «jadem» i «jedzeniem».

Jedzenie - klatka, nadepnąc się na którą, osobnik uzupełnia zdrowie.

Jad - klatka, nadepnąc się na którą, osobnik traci zdrowie.

Osobnik - obiekt posiadający genom, mający parametry punktów zdrowia, współrzędnych osi koordynat i ID. Krótko - bot.

Opis projektu

Projekt reprezentuje zastosowanie algorytmu genetycznego, na przykładzie uczenia się «botów» (dalej «osobniki»), czyli znalezienie lepszego genomu (najskuteczniejszego rozwiązania problemu) w wyniku doboru naturalnego, ewolucji osobników.

W praktyce program wygląda następująco, przy założeniu że zaczynamy symulację od początku: tworzy się nowy zbiór botów, które mają generowany losowo genom i 50 punktów zdrowia; osobniki przywiązują się do klatek sgenerowanego wcześniej pola, na którym są również klatki z jadem i jedzeniem. Po ustawieniu parametrów zaczyna się iteracja, w czasie trwania której boty zachowują się według instrukcji zapisanych w ich genomie (o algorytmie genomu bedzie poniżej). Po pewnym czasie zostają najlepsze osobniki, mające najbardziej dostosowany do określonych przez program warunków genom. W taki sposób wykonują się dalsze iteracj, z czasem trwania których boty co raz lepiej radzą sobie z wytrwaniem w zasymulowanym świecie.

Dany algorytm świetnie sprawdza się w uczeniu się maszynowym i częściowo jest podobny do sieci neuronowych, chociaż nie jest na tyle skuteczny: nie może rozwiązywać złożone problemy, wymaga wiecej «prób» na porządkowanie algorytmu i jest bardziej intuicyjny, czyli nie uogólnia dane a stara się je przewidzieć. Zaletą algorytmu genetycznego jest dość prosta idea na której on się bazuje: «To co słabe musi umrzeć». Więc do rozwiązania niezłożonych problemów opłaca się go stosować

P.S: Po dużym czasie pracy nad programem największym problemem było znalezienie bilansu pomiędzy różnymi metodami botów, które mogą wpływać na stan tego samego bota i jego otoczenia. Zaprogromowanie tych wszystkich «osobników» i «świata» w którym się znajdują jest dość banalne. Lecz ustawienie takich warunków, żeby pokolenie jak najdłużej żyło i się rozwijało, wymaga obserwacji zachowania botów i właśnie w tym dostrzegłem «piękno» tego algorytmu.

Opis formatu danych wejściowych/wyjściowych

Dane wejściowe (początkowe):

UWAGA! PARAMETRY POCZĄTKOWE JUŻ MAJĄ DOMYŚLNE WARTOŚCI, DLA ICH ZMIANY NALEŻY ZMIENIĆ JE BEZPOŚREDNIO W KODZIE (W PLIKU «algorytm.cpp»)

int:

- Wymiary pola «w * k» (domyślnie 25x25);
- Ilość osobników «bots» (domyślnie 100), jadu «poisonNumber» i jedzenia «goodNumber» (domyślnie po 150);
- ilość iteracji «p» (podaje się w konsoli).

double:

- Czas odświeżania pola w konsoli «roundTime» (w milisekundach, domyślnie 0.1 s).

Dane wyjsciowe:

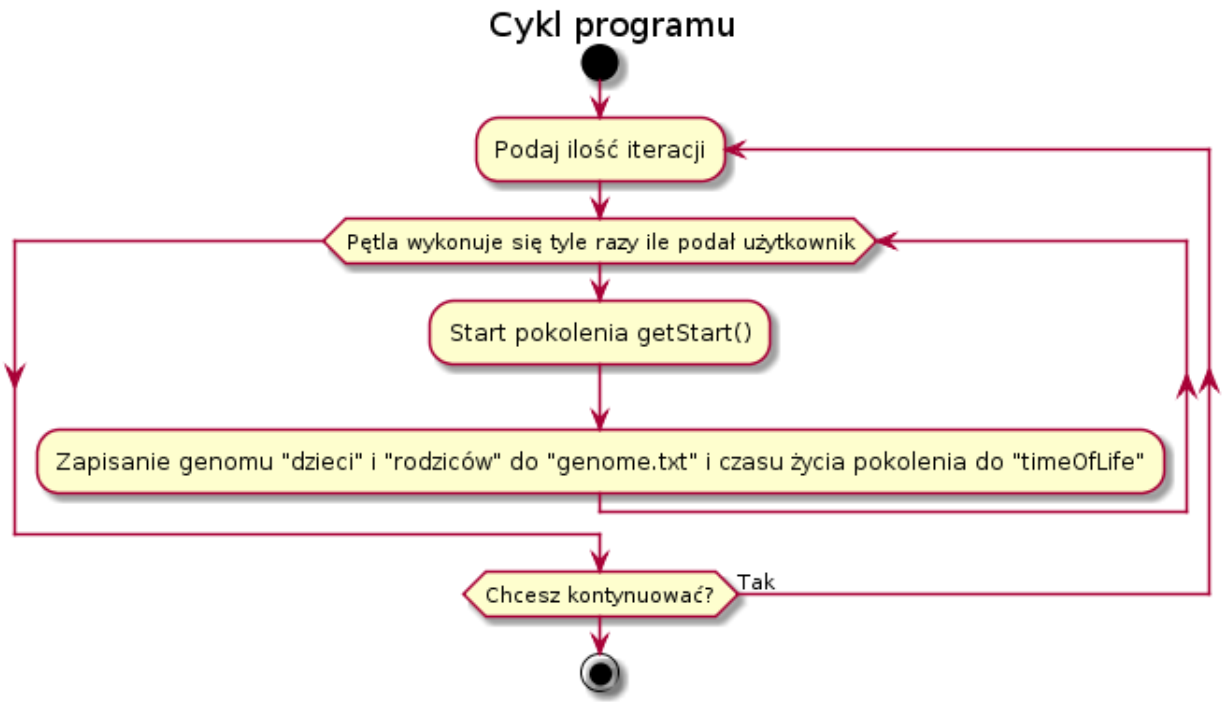
konsolowe:

- Wyświetlanie listy osobników, genomu i pola odpowiednio funkcjami «showBotsList()», «showGenome()», «showKlatka();
- Wyświetla ID osobnika, który obecnie coś robi (move, atack, catchEatHeal, look, moveGenomePointer);
- Numer pokolenia, czas życia, genom 10% osobników, które będą się krzyżować są wyświetlane za pomocą funkcji «cycle()» w końcu każdego cyklu.

plikowe:

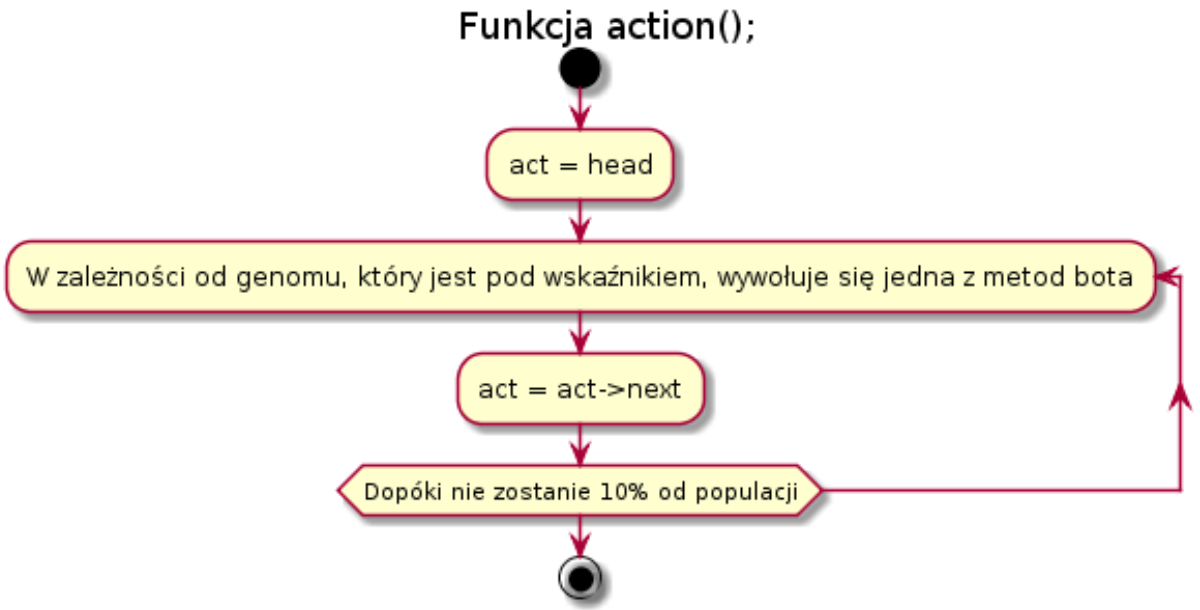
- Genom nowego pokolenia po skrzyżowaniu jest zapisywany do pliku «genome.txt» w folderze programu;
- Lista czasu życia każdego pokolenia w pliku «TimeOfLife».

Szkielet programu

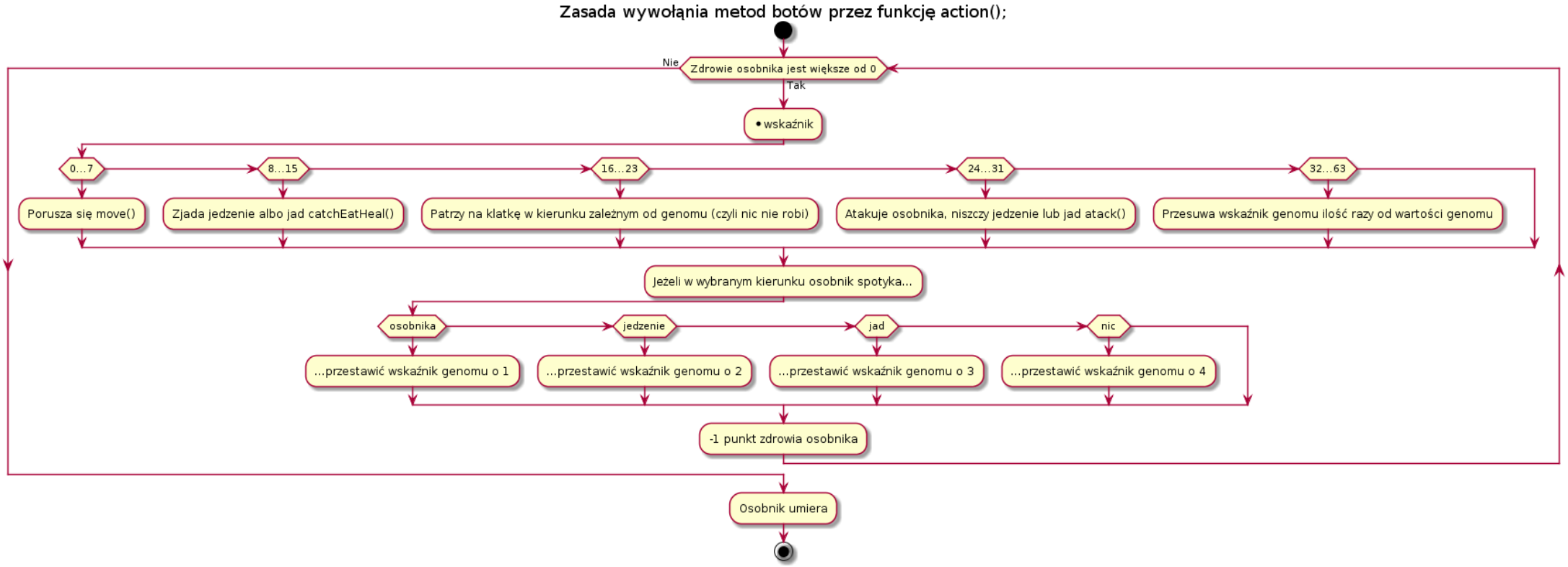


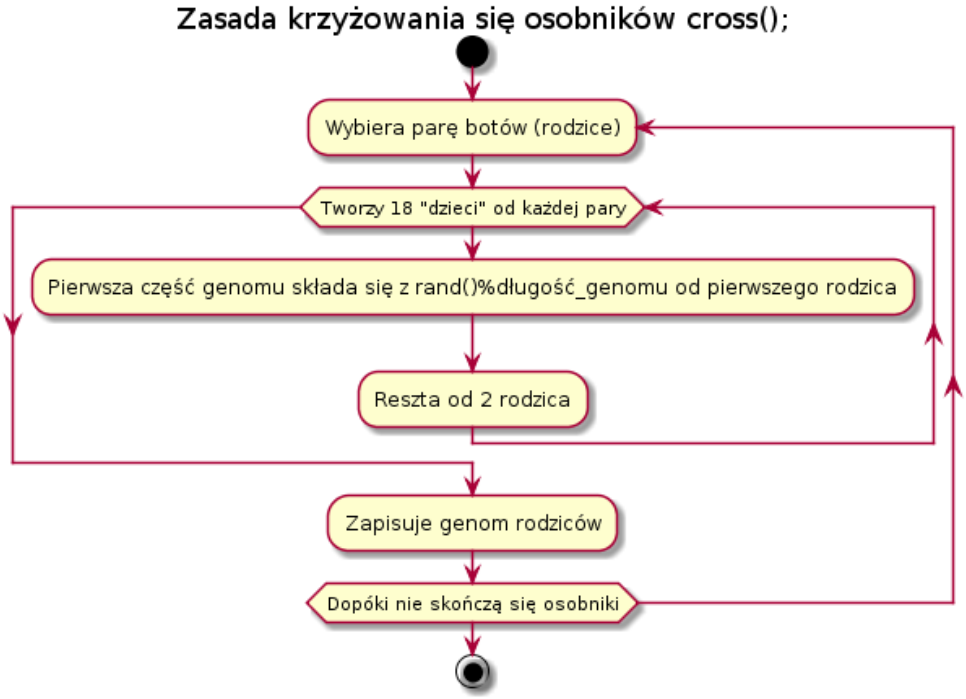
Główne funkcje:

- getStart(): wywołuje funkcje tworzące i usuwające listę botów i tablicę klatek, wywołuje action() i po zakończeniu doboru cross();



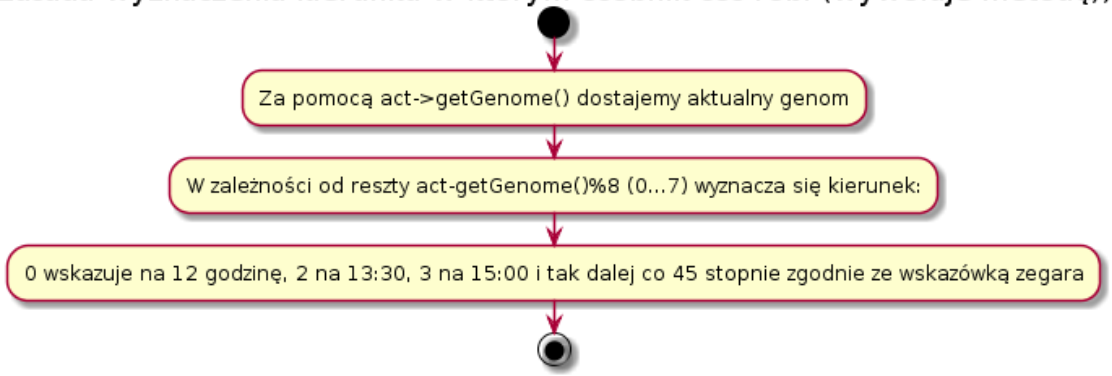
- action(): odpowiada za wywołanie odpowiedniej metody zachowania osobnika w zależności od jego aktualnego genomu;





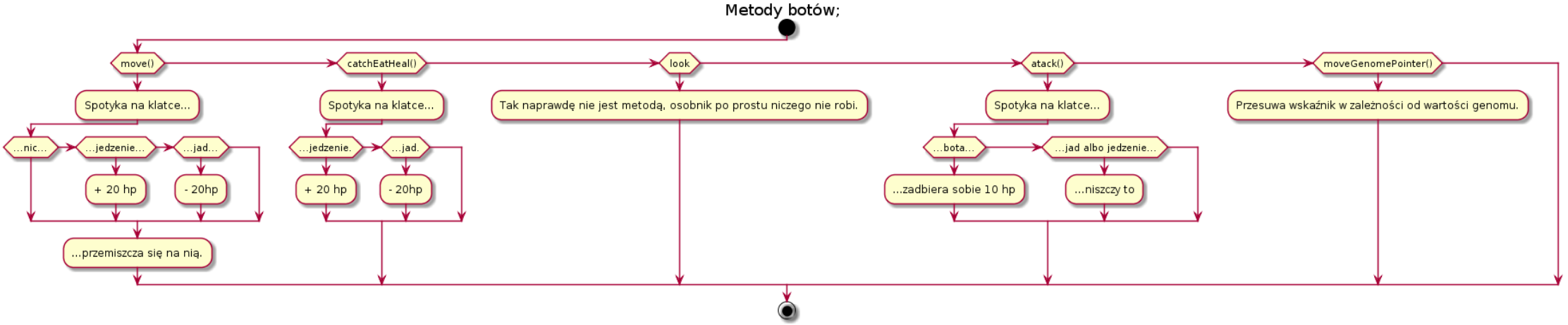
- cross(): odpowiada za krzyżowanie się i mutację «rodziców», zapisanie genomu nowego pokolenia do pliku «genom.txt»;

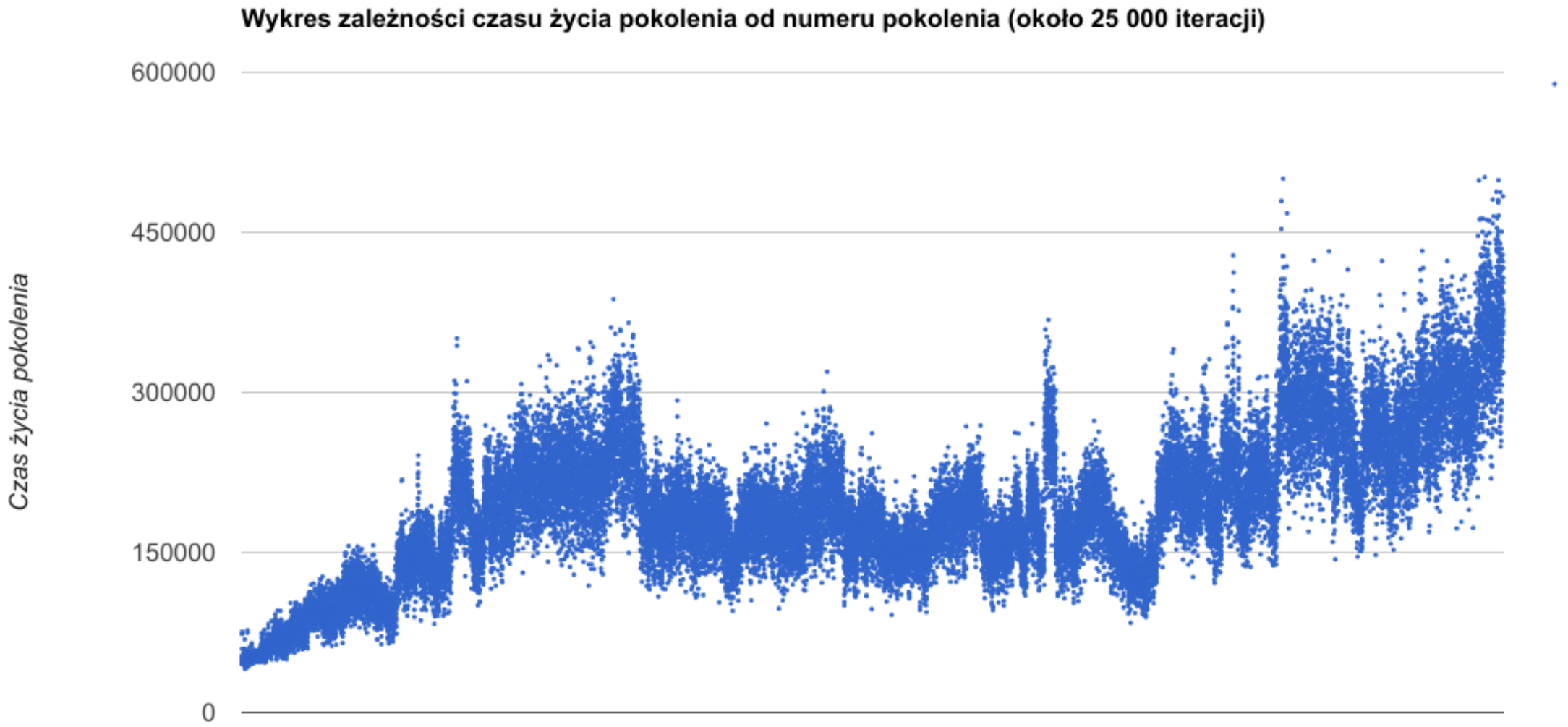
Zasada wyznaczenia kierunku w którym osobnik coś robi (wywołuje metodę);



- direct(): zapisuje współrzędne klatki, na którą osobnik wywołuje metodę;

Metody botów;




Wykres zależności czasu życia pokolenia od numeru pokolenia (około 25 000 iteracji)
600000
450000
300000
150000
0
Czas życia pokolenia