Załącznik

Projekt 2 – inspiracje, przykładowe tematy

Inspiracje

Linki z różnymi pomysłami na projekty

- Różne projekty: https://www.datacamp.com/blog/machine-learning-projects-for-all-levels
- Projekty RL: https://neptune.ai/blog/best-reinforcement-learning-tutorials-examples-projects-and-courses
- Projekty związane z chatbotami: https://www.datacamp.com/blog/5-projects-you-can-build-with-generative-ai-models https://www.datacamp.com/courses/building-chatbots-in-python
- Tekstowe: https://www.datacamp.com/tutorial/nlp-project-ideas-for-all-levels
- Zrobienie żyjącej populacji Sieci Neuronowe/Algorytmy genetyczne/NEAT: https://www.youtube.com/watch?v=N3tRFayqVtk
- Gry/środowiska/światy sterowane przez LLM: https://www.youtube.com/watch?v=Se6KFn1Nni4&ab_channel=MatthewBerman https://www.youtube.com/watch?v=CaGqEBjsJtA&ab_channel=BrendanGalea https://github.com/northern-lights-province/calypso-aiide-artifact

---

Poniższe tematy były proponowane studentom także rok temu. Można je wybrać, ale zalecane jest szukanie swoich własnych. Proszę też zwrócić uwagę na to, ze opisy tych tematów mogą nie przystawać do aktualnych wymagań projektu nr 2.

Temat 1

Szukanie drogi w labiryncie

Osoby wybierające ten temat, muszą go opracować bardzo porządnie, bo temat był omawiany na zajęciach, więc może wydawać się bardziej oswojony i łatwiejszy. Przypomnienie problemu:

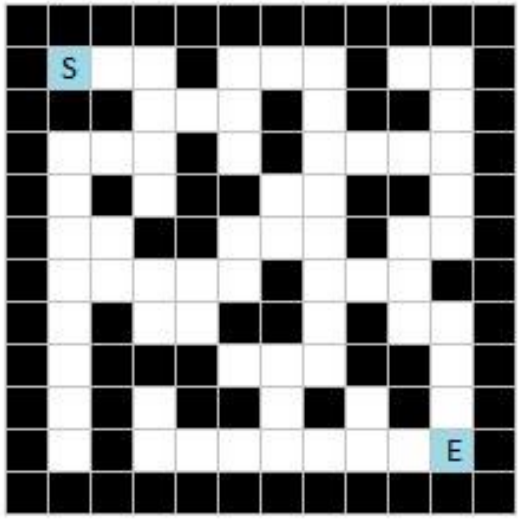
Dany jest prostokątny labirynt składający się z kwadratowych pól. Niektóre pola są puste i można po nich chodzić (kratki białe), a niektóre pola są ścianami i nie można na nie wchodzić (kratki czarne). Można założyć, że labirynt jest otoczony ramką ścianek, tak aby nie dało się wyjść poza jego granice.

Po labiryncie chodzimy przeskakując z pola na jedno z sąsiednich pól (po lewej, prawej, u góry lub na dole). Nie są legalne skoki na ukos.

W lewym górnym rogu labiryntu znajduje się pole startu (S), a w prawym dolnym rogu pole wyjścia (exit, E). Czy istnieje ścieżka z S do E o określonej maksymalnej długości?

Instancja problemu:

Labirynt 12x12:



Maksymalna liczba kroków, długość ścieżki: Limit =30

Oczywiście w ramach projektu należy powymyślać kilka labiryntów do testów, np. o wielkościach 10x10, 20x20, 30x30 i sensownych limitach.

Podpowiedzi do rozwiązania algorytmem genetycznym i rojem cząstek:

A. Pierwszy sposób kodowania przedstawiliśmy na zajęciach. **Chromosomy to sekwencje ruchów** Lewo, Prawo, Góra, Dół zakodowane jako cztery liczby np. 0, 1, 2, 3. Ustalamy długość chromosomu na Limit (w razie szybszego znalezienia rozwiązania ignorujemy ostatnie ruchy).

   1) Dla powyższego kodowania omówiliśmy **brutalną funkcją fitness**. Funkcja wykonuje ruchy chromosomu zaczynając od start. Gdy wejdzie na ścianę to kończy wykonywanie ruchów i zwraca odległość kolizji (x,y) od wyjścia E, np. za pomocą metryki Manhattan z minusem: $-(|x - x_{exit}| + |y - y_{exit}|)$.
   Postprocessing: po zwróceniu rozwiązania przez A.Gen. trzeba sprawdzić czy ostatnie ruchy zostały wykorzystane, czy nie. Jeśli nie, to usuwamy je.

   2) Innym sposobem była funkcja **fitness odbijająca się od ściany**. Funkcja ta, w przypadku ruchu, który chce wejść na ścianę, ignoruje ten ruch i stoi w miejscu. Funkcja ta porusza się tak długo, aż zabraknie ruchów lub trafi na wyjście, i zwraca odległość od wyjścia.
   Postprocessing: po zwróceniu rozwiązania przez A.Gen. trzeba sprawdzić czy ostatnie ruchy zostały wykorzystane, czy nie. Jeśli nie, to usuwamy je. Dodatkowo usuwamy wszystkie ruchy, które wchodziły na ściany i nie zostały wykorzystane.

   3) Najbardziej przyjaznym rodzajem funkcji może być **fitness zmieniająca kierunek ruchu**. Oczywiście fitness nie ma prawa edytować chromosomów, ale można

wprowadzić wewnętrzną zasadę o takiej treści: jeśli ruch chce wejść na ścianę, to wykonaj ruch w inną stronę (np. zmień zgodnie z ruchem wskazówek zegara). *Postprocessing: po zwróceniu rozwiązania przez A.Gen. trzeba sprawdzić czy ostatnie ruchy zostały wykorzystane, czy nie. Jeśli nie, to usuwamy je. Dodatkowo naprawiamy wszystkie ruchy wchodzące na ścianę w rozwiązaniu, zamieniając je na ruchy legalne, które wykonała fitness (przekręcanie zgodnie z ruchem wskazówek zegara).*

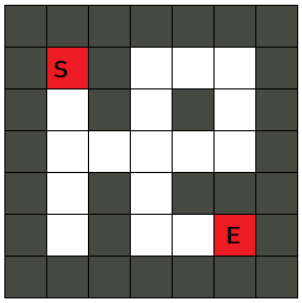
4) Są jeszcze podejścia podobne do powyższych, ale naliczające punkty karne lub bonusowe. Na ocenę może mieć wpływ:
    - ile ruchów wykonaliśmy zanim znaleźliśmy wyjście
    - czy wchodzimy na pola, z których właśnie wyszliśmy (zawracanie)
    - czy wchodzimy na pola, na których kiedyś wcześniej byliśmy (pętle)
    - uderzenia w ścianę – są legalne (jak w podejściu 2), ale dają punkty karne

    Dobre skomponowanie funkcji z tych kilku komponentów może być trudne. Pamiętajmy, że nadrzędnym celem powinno być dojście do wyjścia. Odległość od wyjścia musi być mocno punktowana.

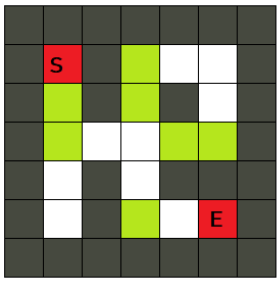
B. Nieco zmieniony sposób kodowania chromosomów: zamiast kodować wszystkie ruchy ścieżki, skupmy się tylko na **skrzyżowaniach**. Są to pola w labiryncie, które graniczą z 3 lub 4 pustymi polami. Namierzamy wszystkie skrzyżowania w labiryncie, zapisujemy je sobie w określonej kolejności, a następnie tworzymy **chromosom o długości równej ilości skrzyżowań**. Fitness wychodzi ze startu i idzie ścieżką aż trafi na skrzyżowanie, wówczas odczytuje z chromosomu, w którą stronę iść (trzy wybory: skręć w lewo=0, skręć w prawo=1, idź prosto=2). W przypadku kolizji można przyjąć jedną z trzech strategii przedstawionych powyżej. Taka strategia odchudza chromosomy i rozwiązanie będzie znajdowane szybciej, wymaga jednak preprocessingu i znalezienia wszystkich skrzyżowań.

C. Jeszcze inny sposób kodowania chromosomów (nieco egzotyczny, ale ze sporym potencjałem): **kodujemy ścieżkę jako zamalowanie białych pól labiryntu**.

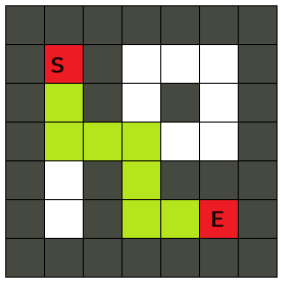
   Zliczamy wszystkie puste pola labiryntu i ustawiamy je rzędami w chromosomie. Przykładowo taki labirynt:



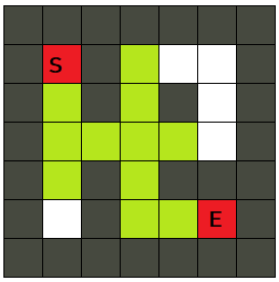
   Ma 16 białych pól. Więc chromosom jest 16-bitowy.
   Jeśli w chromosomie na danym polu jest 0, to znaczy, ze nie przechodzimy przez to pole, jeśli jest 1 to znaczy, że przechodzimy. Przykłady trzech chromosomów z zaznaczonymi ścieżkami:



1001101001100010



0001001110001011



1001101111011011

Jak widać, pierwsza ścieżka nawet nie jest ścieżką i powinna być źle oceniona. Druga ścieżka jest idealna. Powinna mieć najwyższą ocenę. Trzecia jest dobra, ale ma rozgałęzienia i być może nadmiar wykorzystanych pól.

Jak oceniać chromosomy w takim kodowaniu? Musimy zastosować zupełnie inne taktyki. Nie wykonujemy tu symulacji ruchów, nie wchodzimy na ściany. Zamiast tego trzeba ocenić to, jak bardzo ścieżka przypomina poprawną ścieżkę. Funkcja fitness powinna namalować ścieżkę w labiryncie patrząc na zera i jedynki z chromosomu, a następnie ocenić namalowaną ścieżkę:

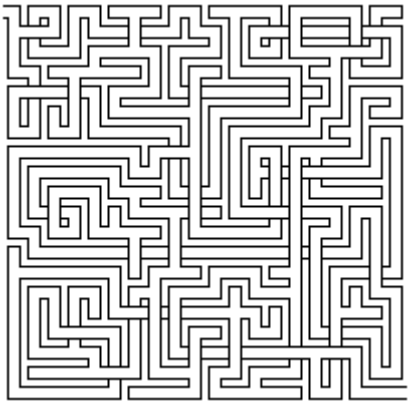
- Czy ścieżka łączy się ze START?
- Czy ścieżka łączy się z EXIT?
- Czy ścieżka jest spójna? Jeśli nie, to z ilu kawałków się składa?
- Z ilu pól składa się ścieżka? Czy ścieżka ma rozgałęzienia?

W sensownym sposób trzeba ustalić punktację i ocenianie, tak żeby premiowane były dobre właściwości a karane złe właściwości ścieżki.

D. Rój cząstek może korzystać z funkcji fitness wykorzystywanych w algorytmie genetycznym. W podejściu C. skorzystamy z algorytmu BinaryPSO. W podejściu A. i B, trzeba by wykorzystać jakieś podejście hybrydowe z zaokrąglaniem liczb rzeczywistych lub Integer PSO.

E. W Labiryncie dobrze sprawdzi się algorytm A* (A Star). Możne spróbować porównać jego efektywność z algorytmem genetycznym i rojem cząstek. Dla małych labiryntów będzie prawdopodobnie dużo szybszy. Przeszukiwanie labiryntu niczym grafu metodami W Głąb i Wszerz też powinno przynieść dobre rezultaty.r

F. Jeśli zależy ci na Max punktów (15 p), to projekt ten trzeba rozwiązać bardzo starannie, trochę powyżej oczekiwań, lub wprowadzić w nim jakieś innowacyjne zmiany np. labiryntu o polach kwadratowych/trójkątnych/sześciokątnych lub szukanie jakichś zasobów w labiryncie. Można wprowadzić mostki / tunele np. tak jak tutaj:

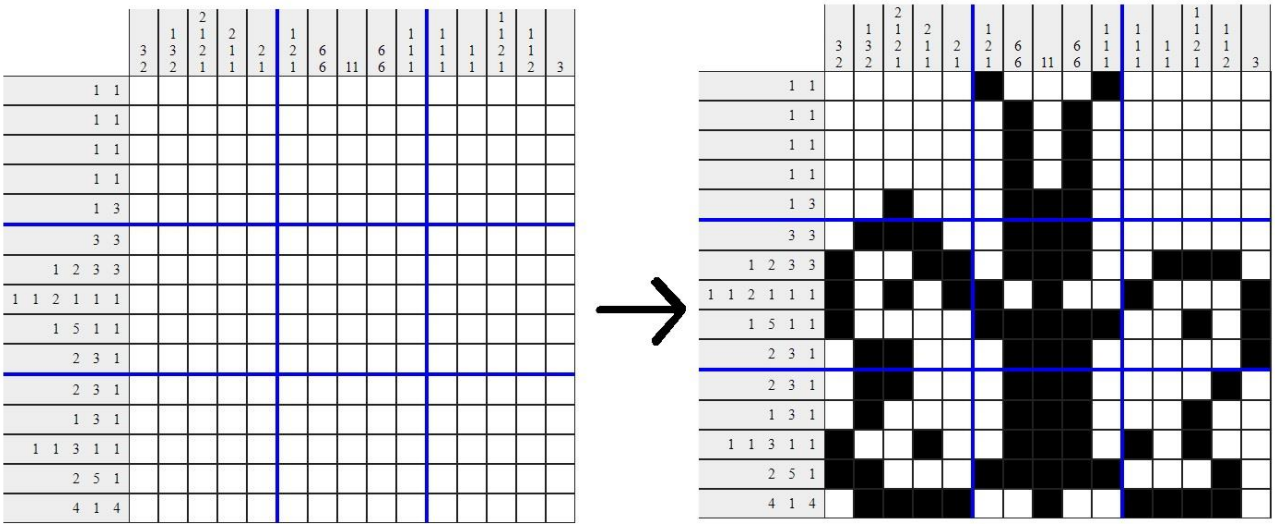


Temat 2

Rozwiązywanie nonogramów

Nonogram (zwany też obrazkiem logicznym) to rodzaj zagadki, w której należy zamalować niektóre kratki na czarno tak, by powstał z nich obrazek. By odgadnąć, które kratki należy zamalować, należy odszyfrować informacje liczbowe przy każdym wierszu i kolumnie kratek obrazka. Przykładowo, jeśli przy wierszu stoi "2 3 1", to znaczy, że w danym wierszu jest ciąg dwóch zamalowanych kratek, przerwa (przynajmniej jedna biała kratka), trzy zamalowane kratki, przerwa, jedna zamalowana kratka. Umiejscowienie zamalowanych ciągów nie jest wskazane.

Mając daną nieuzupełnioną planszę, pytamy: czy istnieje rozwiązanie dla tej zagadki (tzn. odpowiednie zamalowanie pól bez żadnych sprzeczności)? Poniżej przedstawiono instancję problemu (po lewej) i jej rozwiązanie (po prawej). Rozważ, jak znajdować rozwiązanie za pomocą algorytmu genetycznego, PSO lub innych algorytmów.



Podpowiedzi do rozwiązania:

- Zanim przeczytasz odpowiedzi, zastanów się jakbyś to rozwiązał(a) samodzielnie!

- Prosty pomysł na algorytm genetyczny: chromosomy mają tyle bitów, ile pól ma plansza. 1 = pole zamalowane, 0 = niezamalowane. Dla obrazka powyżej: 225 bitów.

- Funkcja fitness w najprostszej postaci: policz ile reguł na brzegu planszy jest spełnionych przez rozwiązanie z chromosomu. Funkcja ta jednak jest dość prymitwna i słabo będzie naprowadzała na rozwiązania. Warto rozpatrzyć różny stopień spełnienia reguły np.
    - czy w danym rzędzie (kolumnie) jest odpowiednia liczba kratek
    - czy w danym rzędzie (kolumnie) jest odpowiednia liczba bloków kratek
    - jak bardzo bloki różnią się od tych zadanych w regule

  Mile widziany jest „tuning" takiej funkcji i jej odpowiednie dopasowanie.

- Dla powyższego pomysłu można też uruchomić algorytm BinaryPSO i zobaczyć czy działa szybciej niż algorytm genetyczny.

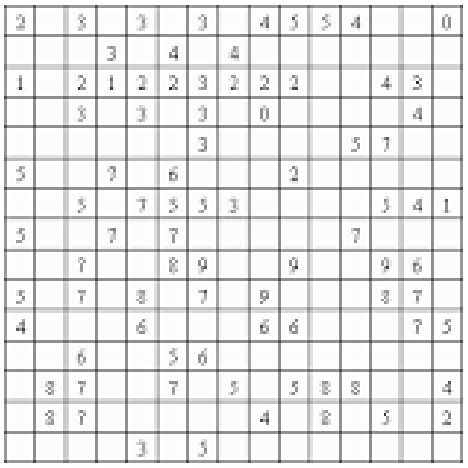
- Inny pomysł na chromosomy warty przetestowania: bloki z reguł są ustalone i wkładamy je na planszę w całości. Chromosom koduje w jakie miejsca wkładać bloki zamalowanych kratek. Chromosom ma zatem *m* licz naturalnych, gdzie *m* to liczba bloków z wszystkich reguł. To podejście ma większą szanse na zadziałanie. Nie wiadomo czy dla tego podejścia znajdzie się też jakaś wersja PSO (może Integer PSO?).

- Inna wersja tego problemu: Nonogramy kolorowe. Można rozpatrzyć zamiast czarnobiałych.

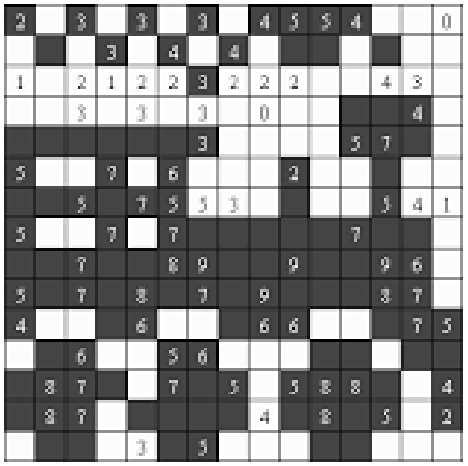
Temat 3

Rozwiązywanie Fill-a-pixów

Fill-a-pix (zwane też mozaikami, Nurie, Nampre) to rodzaj łamigłówki podobnej do nonogramów (i trochę do sapera). Mamy planszę, na której musimy zamalować kratki na czarno, tak aby utworzyły obrazek. Niektóre kratki mają wewnątrz liczbę, która informuje, ile kratek na czarno powinno być w sąsiedztwie. Maksimum sąsiadów to 9: 8 dookoła kratki i 1 to sama kratka, minimum to 0. Niektóre pola mogą być puste – nie informują o liczbie czarnych kratek w sąsiedztwie, mimo że mogą je mieć.



Po prawej znajduje się przykład nierozwiązany, a pod spodem rozwiązanie: obrazek z lokomotywą.



Czy da się takie mozaiki rozwiązać algorytmami genetycznymi, PSO lub innymi algorytmami?

Zanim przeczytasz podpowiedzi, pomyśl jakbyś to rozwiązał(a).

Podpowiedź:

- Oczywiście można zastosować taktykę z tematu 2 o nonogramach.
- Chromosom to zamalowanie obrazka 1 = pole zamalowane, 0 = niezamalowane. U nas to 225 bitów. Fitness sprawdza ile reguł liczbowych jest spełnionych. Bardziej zaawansowana wersja fitness może też sprawdzać jak bardzo niespełnione są reguły (czy dużo kratek brakuje, albo czy jest ich dużo za dużo, a może różnice są niewielkie?)

- Dla powyższego pomysłu można spróbować też uruchomić Binary PSO.

- Nie wiadomo czy istnieje inne, lepsze kodowanie chromosomów lub funkcje fitness. Jeśli wpadniesz na coś, koniecznie poeksperymentuj ze swoją strategią.

Temat 4

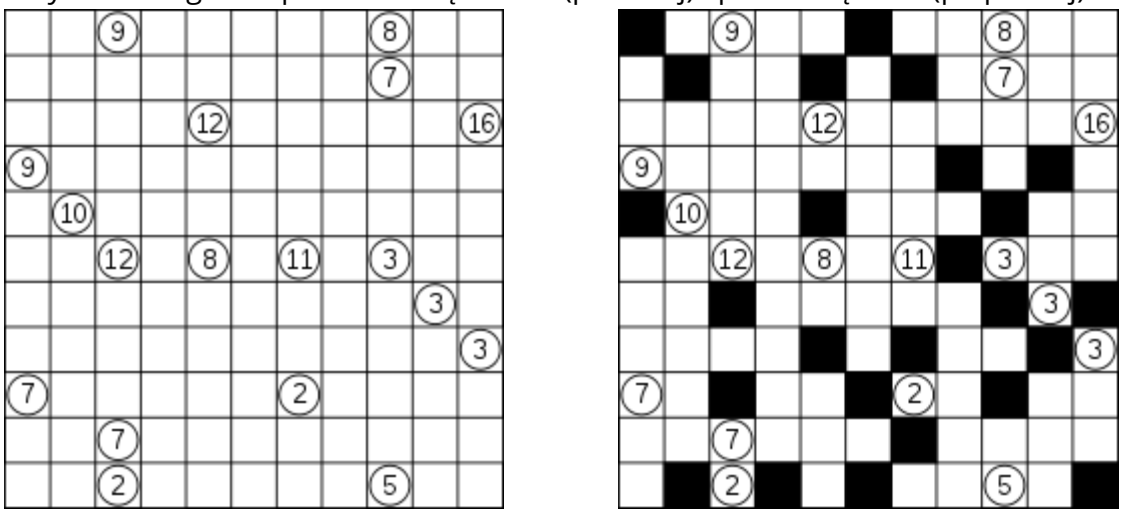
Rozwiązywanie łamigłówki Kuromasu

Łamigłówka Kuromasu składa się z prostokątnej planszy, w której mamy pola puste i oznaczone liczbami. Niektóre puste pola trzeba zamalować ściankami, tak aby pola z liczbami

„widziały” tyle kratek (białych lub z liczbami) w pionie i poziomie ile wynosi liczba w tej kratce. „Wzrok” kończy się na brzegu planszy lub namalowanej ściance.

Więcej na Wikipedii: https://en.wikipedia.org/wiki/Kuromasu

Przykład łamigłówki przed rozwiązaniem (po lewej) i po rozwiązaniu (po prawej).



Czy da się tę łamigłówkę rozwiązać algorytmami genetycznymi, PSO lub innymi algorytmami?

Zanim przeczytasz podpowiedzi, pomyśl jakbyś to rozwiązał(a).

Podpowiedź:

- Chromosom dla algorytmu genetycznego może był tak duży ile mamy białych kratek. Pole pozostawione jako niezamalowane ma wartość 0, a zamalowane 1.
- Funkcja fitness w wersji bazowej może sprawdzać ile kratek z liczbami ma spełnioną „regułę widzenia” odpowiedniej liczby kratek. W nieco bardziej zaawansowanym podejściu, fitness może oceniać jako bardzo źle ograniczona jest kratka z liczbą. Np. jeśli w kratce jest 12 i ta 12 widzi tylko 3 pola, to można dać w tym przypadku więcej punktów karnych.
- Warto spróbować dla powyższych funkcji uruchomić algorytm Binary PSO.

- **Uwaga!** Jeśli proponowane tutaj łamigłówki Ci nie pasują, możesz poszukać innych ciekawych np. tutaj: https://en.wikipedia.org/wiki/Nikoli_(publisher)

Temat 5

Generowanie labiryntów

Nasze bioinspirowane algorytmy, genetyczny i rój cząstek, mogą też generować pewne struktury, szukając takiej, która ma najlepsze właściwości.
Pomysł jest następujący: generujemy labirynty składające się z kwadratowych pól: pól pustych (do chodzenia) i ścianek. Rozwiązaniem/Chromosomem będzie zatem sam labirynt, każde jego pole odpowiada dwóm bitom (0 = puste, 1 = ścianka).

Funkcja fitness ocenia wygenerowany labirynt wg różnych kryteriów. Można zademonstrować działanie różnych funkcji fitness, które mają różne kryteria. Przykłady takich kryteriów:

- Stosunek liczby ścianek do liczby pustych pól w labiryncie.
- „Granulacja" ścianek. Czy ścianki są pojedynczymi kratkami, czy zbijają się w klastry?
- Czy ścianki tworzą podłużne linie np. 1x4? Czy mogą tworzyć zbite struktury np. 2x2, 4x4?
- Tak samo dla pustych pól. Czy tworzą wąskie ścieżki? A może dozwolone są też większe „sale" w labiryncie.
- Liczba skrzyżowań. Duża? Mała?
- Liczba zaułków czyli białych kratek otoczonych z 3 stron ścianką.
- Czy do każdego miejsca w labiryncie da się dojść? Są obszary niedostępne?
- Liczba pętli w labiryncie.
- Trudność labiryntu mierzona np. jako długość rozwiązanie + liczba skrzyżowań, które minęliśmy.
- Poziomość vs Pionowość: jakie ściany mają przeważać.

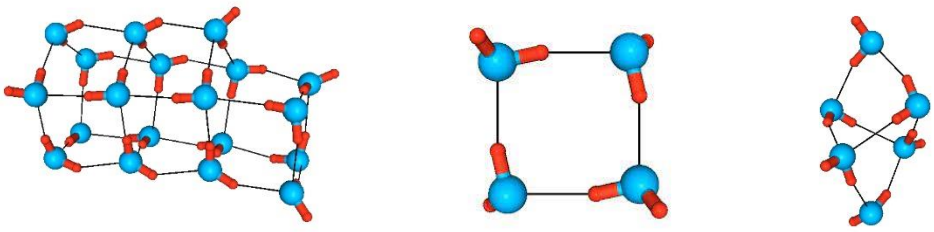
Oprócz algorytmu genetycznego, warto spróbować zastosować algorytm BinaryPSO.

Warto zaprezentować różne generowane labirynty, dla różnych wersji funkcji fitness.
Mile widziane graficzne reprezentowanie labiryntów.

Temat 6

Istnienie i stabilność klastra cząstek

Problem pochodzi z dziedziny fizyki molekularnej/chemii. Pod względem zrozumienia materiału może być trochę odstraszający, ale pod względem technicznym/programistycznym jest stosunkowo prosty.



Mamy układ cząstek (atomów lub molekuł), które oddziałują ze sobą tworząc bardziej skomplikowaną strukturę lub tzw. klaster. Oddziaływanie między każdą parą cząstek można dość dobrze modelować m.in. potencjałem Morse'a, który ma prostą postać analityczną i jest funkcją odległości między tymi cząstkami. Całkowita energia wewnętrzna takiego klastra będzie sumą energii oddziaływania każdej pary cząstek (po wszystkich możliwych parach) i będzie funkcją 3N współrzędnych (gdzie N-liczba cząstek tworzących klaster, każda cząstka ma trzy współrzędne przestrzenne x, y, z). Żeby taki układ mógł zaistnieć, to hiperpowierzchnia

całkowitej energii (suma energii wszystkich par cząstek) musi mieć minimum energetyczne dla jakiejś konfiguracji przestrzennej tego układu. Zadanie polega więc na znalezieniu minimum energii tego układu.
Bardzo pomocny będzie tutaj artykuł: https://inf.ug.edu.pl/~gmadejsk/io-pliki-2021/Art_2a.pdf, który zalecam przejrzeć. Warto przeczytać chociaż wstęp.

Z tego artykułu można się dowiedzieć, że potencjał Morse'a pomiędzy dwiema cząstkami, nr *i* oraz *j,* można zamodelować według wzoru:

$$V_{ij} = e^{\rho_0 \cdot (1 - r_{ij})} (e^{\rho_0 \cdot (1 - r_{ij})} - 2)$$

Gdzie r_{ij} to odległość (zwykła, euklidesowa) pomiędzy cząsteczkami nr *i* oraz *j*, natomiast ρ_0 jest parametrem regulującym stopień przyciągania dwóch cząstek i należy go dopasować do typu cząstek (np. jeśli mamy do czynienia z potasem to $\rho_0 = 3.17$ a dla niklu $\rho_0 = 3.96$).

Potencjał Morse'a dla klastra to suma potencjałów wszystkich możliwych par cząstek, czyli:

$$V_M = \sum_{i<j} V_{ij}$$

To jest funkcja, której minimum należy szukać. Gdy znajdziemy minimum, to mamy potencjalnie stabilne ułożenie cząstek, które tworzy klaster (np. kryształ z atomów).

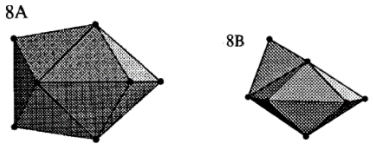
Przekładając na język algorytmu genetycznego. Chromosom to 3N liczb (współrzędnych x,y,z) dla N cząstek. A funkcja fitness to V_M (być może z dodanym sztucznie minusem). Ograniczenia dla x,y,z można ustawić na przedział [0,2], choć można poeksperymentować z rozszerzeniem go. Można wykonać kilka eksperymentów dla różnych ρ_0 – zalecam sprawdzić dla działanie programu dla wartości 3, 6, 10, 14. Ile cząstek wziąć? Tutaj zalecam zacząć eksperymenty od N=5 i zwiększać N co jeden, patrząc czy wyniki wychodzą też dobre dla większych N.

Skąd mam wiedzieć, że program działa?
W podanym wyżej linku do artykułu naukowego znajduje się tabelka z wynikami kilku naukowców. Nasze wyniki powinny wychodzić podobne do wyników tych badaczy.

	Point group	n_{nn}	E_{strain}	ρ_0=3.0	ρ_0=6.0	ρ_0=10.0	ρ_0=14.0
5A	D_{3h}	9	0.000	**−9.299 500**	**−9.044 930**	**−9.003 565**	**−9.000 283**
6A	O_h	12	0.000	**−13.544 229**	**−12.487 810**	**−12.094 943**	**−12.018 170**
7A	D_{5h}	16	0.062	**−17.552 961**	**−16.207 580**	**−15.956 512**	**−15.883 113**
8A	D_{2d}	18	0.006	**−22.042 901**	−19.161 862	−18.275 118	−18.076 248
8B	C_s	19	0.062		**−19.327 420**	**−18.964 638**	**−18.883 688**
9A	D_{3h}	21	0.002	**−26.778 449**	−22.330 837	−21.213 531	−21.037 957
9B	C_{2v}	22	0.062	−26.037 771	**−22.488 044**	**−21.975 747**	**−21.884 483**
10A	D_{4d}	24	0.002	−31.519 768	−25.503 904	−24.204 958	−24.031 994
10B	C_{3v}	27	0.694	**−31.888 630**	**−27.473 283**	**−26.583 857**	**−26.132 735**
11A	D_{4d}	34	10.374	**−37.930 817**	−28.795 153[4]	−23.666 072[5]	
11B	C_{2v}	32		−37.891 674[1]			
11C	C_{2v}	31	0.792		**−31.521 880**	**−30.265 230**	−29.588 130[1]
11D	C_s	30					**−29.596 054**
11E	C_2	30	0.248	−36.613 190[1]	−30.698 890	−29.808 994	−29.524 398
11F	C_{2v}	29	0.001	−36.697 760	−30.431 713	−29.215 51	−29.037 941
12A	C_{2v}	38		−43.971 339[1]	−35.199 881[1]		
12B	C_{5v}	36	1.704	**−44.097 880**	**−36.400 278**	**−34.366 755**	−33.115 942[1]
12C	C_s	34					−33.199 505
12D	D_{2d}	34	0.346	−41.816 393	−34.838 761	−33.724 155	**−33.332 305**
12E	D_{3h}	33	0.001	−42.121 440	−34.568 002	−33.222 331	−33.038 298
13A	I_h	42	2.425	**−51.737 046**	**−42.439 863**	**−39.662 975**	**−37.258 877**
13B	D_{5h}	37	0.141	−49.998 058[1]	−39.360 710[1]	−37.208 019[1]	−36.790 507

Tabelka znajduje się też pod linkiem:
http://doye.chem.ox.ac.uk/jon/structures/Morse/tables.html

Objaśnienia! Pierwsza kolumna mówi ile cząstek wzięto, na przykład 8A i 8B oznacza, że wzięto 8 cząstek, i otrzymano dwa stabilne klastry A i B o różnej strukturze (struktura opisana jest w kolumnie drugiej jako D_{2d} i C_s, ale nie musimy wnikać co to oznacza). W artykule są nawet rysunki tych klastrów:



Kolumna 3 i 4 nie jest dla nas ważna. Ostatnie cztery kolumny opisują wyniki minimalnego V_M naukowców dla różnych ρ_0 . Jeśli wynik jest pogrubiony, to znaczy że wynik wyszedł idealny (minimum globalne). Jeśli wynik jest niepogrubiony, tzn. że wynik wpadł w minimum lokalne i nie jest do końca dobry (klaster jest metastabilny).

Naszym zadaniem jest stworzenie takiej tabelki jak owi badacze. Czy otrzymamy takie same wyniki? Trochę gorsze? Trochę lepsze?

Dodatkowe wymagania:

- Czy PSO również sobie poradzi z tym wyzwaniem? Zrób dla tego algorytmu oddzielną tabelkę. Porównaj wyniki PSO i algorytmu genetycznego. Który osiągał lepsze wyniki?
- W miarę możliwości spróbuj wykorzystać jakąś paczkę graficzną do zwizualizowania otrzymanych najlepszych klastrów (z minimalnym V_M). Rysunek punktów w 3D. Można próbować dodawać ścianki, ale nie jest to konieczne.

Temat 7

Rozwiązywanie problemu 3SAT

Dana jest formuła logiczna w koniunkcyjnej postaci normalnej o maksymalnie 3 literałach w każdej klauzuli (postać 3CNF), np.

$$\begin{aligned}\varphi \;=\; & (x_1 \vee x_2 \vee x_3) \;\wedge\; (\neg x_1 \vee x_2 \vee x_4) \\ \square \;\wedge\; & (x_2 \vee \neg x_3 \vee \neg x_4) \;\wedge\; (\neg x_1 \vee \neg x_2 \vee x_5) \\ \square \;\wedge\; & (x_1 \vee \neg x_3 \vee x_5) \;\wedge\; (\neg x_3 \vee x_4 \vee \neg x_5) \\ \square \;\wedge\; & (x_1 \vee \neg x_2 \vee \neg x_5) \;\wedge\; (\neg x_1 \vee \neg x_4 \vee \neg x_5) \\ \square \;\wedge\; & (\neg x_2 \vee \neg x_3 \vee \neg x_5) \;\wedge\; (\neg x_1 \vee \neg x_2 \vee \neg x_4)\end{aligned}$$

Formuła φ ma 10 klauzul (nawiasów), w każdym po trzy literały (zmienne), z negacją lub bez. Pomiędzy klauzulami są znaki koniunkcji, tzn. żeby $\varphi \equiv True$, wszystkie nawiasy muszą być

spełnione. W środku każdej klauzuli są znaki alternatywy tzn. żeby nawias dał $True$, przynajmniej jedna zmienna w nim musi dać 1.

Problem 3SAT (3-Satisfiability, 3-Spełnialność) zadaje pytanie: czy istnieje takie podstawienie zmiennych, żeby $\varphi \equiv True$?

Jeśli weźmiemy φ z przykładu to widać, że podstawienie $x_1 = 1,\ x_2 = 1,\ x_3 = 0, x_4 = 0,\ x_5 = 1$ spowoduje, że wszystkie nawiasy dadzą $True$ więc również cała formuła $\equiv True$. Dla powyższej formuły logicznej odpowiedź brzmi „TAK". Istnieją oczywiście formuły niespełnialne, dla których nie istnieje podstawienie spełniające.

Twoim zadaniem jest wykorzystanie algorytmu genetycznego, PSO i być może dodatkowego algorytmu np. DPLL, do próby rozwiązania 3SAT. Odpowiedz na pytania: czy te algorytmy rozwiązują 3SAT i czy robią to szybko. Zrób sensowne porównania (tabelaryczne, wykresowe).

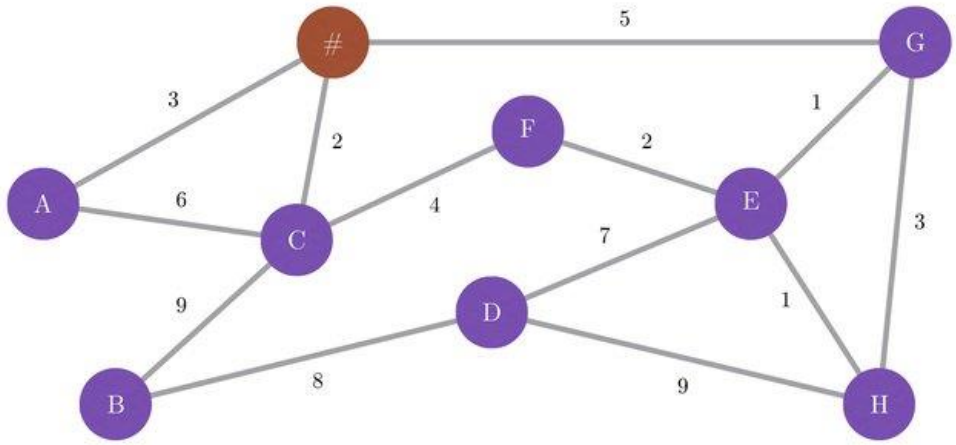
Rozważ wiele różnych formuł, w tym również skomplikowanych np. 400 klauzul, 100 zmiennych. Można je pobrać z Internetu np. tutaj:

- https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html
- https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

Temat 8

Rozwiązywanie problemu komiwojażera lub marszrutyzacji

Problem komiwojażera (travelling salesman problem, TSP), to problem, w którym mamy dany graf z krawędziami etykietowanymi liczbami, np.



i naszym celem jest znalezienie cyklu (ścieżki zaczynającej się i kończącej w tym samym wierzchołku), która odwiedzi wszystkie wierzchołki, a przechodząc przez krawędzie będzie miał jak najmniejszą sumę wag.
Można sobie wyobrazić, że kurier musi objechać wszystkie domy i tak musi wybrać trasę, żeby stracić jak najmniej benzyny/czasu (każda krawędź oznacza ile straci benzyny/czasu jadąc z wierzchołka do wierzchołka).

Problem można też spotkać nie w postaci grafowej, a bardziej analitycznej. Zamiast grafu dostajemy po prostu punkty na płaszczyźnie. Odległości między nimi odpowiadają wagom w grafie.

Nieco ciekawszym problemem, który można wybrać zamiast TSP jest problem marszrutyzacji (https://pl.wikipedia.org/wiki/Problem_marszrutyzacji) , Vehicle Routing Problem, VRP. Z kolei sam VRP ma wiele wariantów, które można wybrać https://en.wikipedia.org/wiki/Vehicle_routing_problem

Jako problem szukania ścieżek warto spróbować ten problem rozwiązać, kilkoma algorytmami:

- Algorytmem genetycznym
- PSO lub ACO (to drugie ponoć działa dobrze, warto przetestować)
- Algorytmami przeszukiwania grafów np. wszerz, w głąb, podejścia heurystyczne, programowanie dynamiczne

Mile widziane porównania czasowe dla różnych algorytmów. Warto tutaj poszukać dużych instancji problemów (grafy z dużą liczbą wierzchołków).

Temat 9

Parkowanie auta – Kontroler rozmyty i inne algorytmy sterujące

Można rozpatrzyć problem parkowania auta i napisać kontroler rozmyty, który to automatycznie zrobi zamiast kierowcy. Dodatkowo można spróbować napisać inne systemy (własny skrypt, reinforcement learning, sieć neuronowa). Na końcu zademonstruj działanie algorytmów na prostej animacji z parkowaniem auta (możesz zapisać jaki video lub gify).

Różne linki, które mogą pomóc:

- https://www.researchgate.net/publication/224687286_Design_of_Fuzzy_Controller_for_Car_Parking_Problem_Using_Evolutionary_Multi-objective_Optimization_Approach
- https://www.academia.edu/41104980/Designing_a_Fuzzy_Logic_Controller_for_Automated_Backward_Parking_Car
- https://arxiv.org/ftp/arxiv/papers/1912/1912.06764.pdf

Temat 10

Samochód autonomiczny – Kontroler rozmyty i inne algorytmy sterujące

Bardziej zaawansowany problem niż poprzedni temat.

Można rozpatrzyć problem autonomicznego sterowania autem (jazda, omijanie przeszkód) i napisać kontroler rozmyty, który to wykona. Dodatkowo można spróbować napisać inne systemy (własny skrypt, reinforcement learning, sieć neuronowa). Na końcu zademonstruj działanie algorytmów na prostej animacji (możesz zapisać jaki video lub gify)

Różne linki, które mogą pomóc:

- https://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=1035&context=me_fac_articles
- https://www.hindawi.com/journals/cin/2016/9548482/

Można skorzystać z paczki Gym / Gymnasium jeśli ograniczymy się do wyścigów po trasie:

https://www.gymlibrary.dev/environments/box2d/car_racing/

https://gymnasium.farama.org/environments/box2d/car_racing/

Temat 11

LunarLander – Algorytmy sterujące lądowaniem

W tym przypadku warto skorzystać z paczki Gym/ Gymnasium:

https://gymnasium.farama.org/environments/box2d/lunar_lander/

https://www.gymlibrary.dev/environments/box2d/lunar_lander/

Chcemy pomyślnie wylądować sondą na księżycu. Napisz program, który to wykona. Do rozpatrzenia są trzy możliwości:

- własny skrypt,
- algorytmy reinforcement learning (w tym także z siecią neuronową)
- kontroler rozmyty (wymaga napisania bazy i reguł)
- strategie metaheurystyczne (PSO lub Algorytm Genetyczny)

Na końcu zademonstruj działanie algorytmów na animacjach (możesz zapisać jaki video lub gify)

Temat 12

Dwunóg – Algorytmy sterujące chodzeniem

W tym przypadku warto skorzystać z paczki Gym/ Gymnasium:

https://gymnasium.farama.org/environments/box2d/bipedal_walker/

https://www.gymlibrary.dev/environments/box2d/bipedal_walker/

Chcemy pomyślnie wylądować sondą na księżycu. Napisz program, który to wykona. Do rozpatrzenia są trzy możliwości:

- własny skrypt,
- algorytmy reinforcement learning (w tym także z siecią neuronową)
- kontroler rozmyty (wymaga napisania bazy i reguł)

Na końcu zademonstruj działanie algorytmów na animacjach (możesz zapisać jaki video lub gify)

Na stronach gymów są i inne ciekawe problemy. Jeśli te tematy Ci nie odpowiadają, wybierz inne!

Temat 13

Gra w czołgi: algorytmy sterujące

Na stronie: https://inf.ug.edu.pl/~gmadejsk/tanks/tanks.html znajduje się gra - strzelanka czołgowa. Poniżej planszy z grą są instrukcje sterowania dla graczy. Zamiast graczy, mogą też grać boty wklejone do okienka (trzeba zaznaczyć ptaszkiem, ze ma grać bot). Cała zabawa polega na tym, by napisać jak najlepszy bot.

Celem tego projektu byłoby:

- Napisanie własnego bota do sterowania czołgiem. Być może konieczne będzie wykorzystanie trygonometrii, by dobrze celować w przeciwnika.
- Granie jako ludzie, zbieranie danych gry z logów i wytrenowanie algorytmu do sterowania czołgiem (np. sieć neuronowa).
- Inne zaawansowane rzeczy: kontroler rozmyty / reinforcement learning?

Instrukcja jak pisać boty, obsługiwać grę i zdobywać dane znajduje się pod linkiem:

https://inf.ug.edu.pl/~gmadejsk/PD-3-tank.pdf

Uwaga, instrukcja zawiera wymagania do projektu z 2019 roku. Wymagania te można w części zignorować, a skupić się na tym jak obsługiwać samą grę.

Temat 14

Tweety: Wojna na Ukrainie

Uwaga: narzędzia do ściągania tweetów są aktualnie zablokowane. Realizacja tematu w 2024 jest trudniejsza.

Wyszukać tweety zawierające słowo #ukraine w całym roku 2022 (może być i inny, mniejszy okres). Tweety można zebrać z określonego miejsca, i o określonym języku.

Tweety można zebrać na przykład z każdego dnia w roku. Np. pobrać po 100 na dzień. Wyszłoby ponad 35 tysięcy tweetów.

Nastepnie robimy standardową analizę tekstu (bag of words, chmury tagów). Warto zrobić analizę sentymentu np. dla poszczególnych tygodni w roku (albo miesięcy). I zobaczyć jak zmieniają się emocje.

Temat 15

Tweety i sondaże wyborcze

Uwaga: narzędzia do ściągania tweetów są aktualnie zablokowane. Realizacja tematu w 2024 jest trudniejsza.

Wyszukać tweety zawierające hasztagi związane z partiami politycznymi np.

- #prawoisprawiedliwosc lub #pis
- #koalicjaobywatelska lub #ko
- Inne partie...

w wybranym dłuższym okresie czasu. Tweety można zebrać z określonego miejsca, i o określonym języku.

Nastepnie robimy standardową analizę tekstu (bag of words, chmury tagów). Warto zrobić analizę sentymentu np. dla poszczególnych tygodni w roku (albo miesięcy). I zobaczyć jak zmieniają się emocje ludzi wobec tych partii. Można tez porównać nasze wyniki w czasie, z sondażami np. ze strony https://ewybory.eu/sondaze/polska/

Czy występuje korelacja?

Temat 16

Tweety i problemy społeczne

Uwaga: narzędzia do ściągania tweetów są aktualnie zablokowane. Realizacja tematu w 2024 jest trudniejsza.

Wyszukać tweety zawierające słowo #węgiel lub #inflacja (itp.) w wybranym dłuższym okresie czasu. Tweety można zebrać z określonego miejsca, i o określonym języku. Nastepnie robimy standardową analizę tekstu (bag of words, chmury tagów). Warto zrobić analizę sentymentu np. dla poszczególnych tygodni w roku (albo miesięcy). I zobaczyć jak zmieniają się emocje ludzi wobec tych tematów.

Można spróbować znaleźć korelację między:

- Sentymentem dla #węgiel a ceną węgla
- Sentymentem dla #inflacja a ceną inflacji

Temat 17

Tweety i rozrywka

Uwaga: narzędzia do ściągania tweetów są aktualnie zablokowane. Realizacja tematu w 2024 jest trudniejsza.

Podobnie jak w poprzednich tematach: można poszukać tweetów, ale tym razem na tematy związane z rozrywką. Np. premierami ciekawych seriali na Netflixie, grami komputerowymi, celebrytami. Emocje dotyczące tych tematów też mogą się zmieniać w czasie (np. przed i po premierze filmu/gry). Warto przeprowadzić takie badania.