PROJEKT z BOWD
Automatyka i Robotyka 2015/16

Skład zespołu:
1. Piotr Mikołajek
2. Artur Hadasz

Opiekun: dr inż. Joanna Kwiecień

1. Wstęp

Celem projektu było przetestowanie działania jednego z algorytmów stadnych w wybranym przez siebie problemie optymalizacyjnym. Wybranym problemem była optymalizacja ruchu platform dostarczających surowce z ich punktów odbioru do punktów, w których są potrzebne. Zastosowano do tego celu algorytm genetyczny połączony ze zmodyfikowanym algorytmem A*.

2. Opis zagadnienia

a) Sformułowanie problemu

W fabryce znajdują się miejsca w których wytwarzane są określone surowce (elementy) a także punkty w których przetwarzane są one dalej. Dostarczanie surowców z punktu odbioru do punktu poboru odbywa się przy pomocy specjalnych platform transportowych, których ograniczenia omówiono poniżej. Należy tak rozdzielić surowce pomiędzy obsługującymi je platformami, aby koszt ich ruchu, liczony jako całkowita przebyta droga, był jak najmniejszy. Czas na dostarczenie wszystkich surowców do punktów docelowych jest ograniczony.

b) Opis zagadnienia

Aby uprościć zadanie przyjęto, że platformy poruszają się po prostokątnej planszy składającej się z kwadratowych pól - na każdym polu znajduje się równocześnie tylko jedna platforma, a platforma porusza się przemieszczając się w każdym ruchu na jedno z sąsiednich pól - nie może ona wyjeżdżać poza planszę. Każdy punkt źródłowy i punkt do dostarczenia surowca również zajmuje dokładnie jedno pole planszy. Dodatkowo część pól jest niedostępnych dla platform. Aby nie nastąpiło zderzenie, platformy nie mogą się "zamieniać miejscami". Osoba, chcąca dokonać optymalizacji ma do dyspozycji pewną liczbę platform (która jest jednym z parametrów algorytmu).

Dodatkowe przyjęte założenia:

1) Jeśli platforma nie przenosi surowców przy danym przyporządkowaniu surowców do platform, jest ona usuwana z planszy i tym samym nie przeszkadza innym platformom w pracy - to założenie jest logiczne i znacznie ułatwia obliczenia.
2) Platformy, z wyjątkiem końcowego czekania nie mogą się zatrzymywać (wprowadzone w celu uproszczenia obliczeń).

c) Model matematyczny

Parametry konkretnej planszy są następujące:

1) $m, n \in \mathbb{N}_+$ - Wymiary planszy
2) $F = \{f_1, f_2, \ldots f_l\}$ - Zbiór pól (pary liczb całkowitych) zabronionych, czyli takich, na których platforma nie może się znaleźć
3) k - liczba surowców
4) Zbiór $Q = \{q_1, q_2, \ldots, q_k\}$ - gdzie $q_i = (x^i_q, y^i_q)$, oznacza pole, w którym znajduje się punkt poboru i-tego surowca przez platformę.
5) Zbiór: $R = \{r_1, r_2, \ldots, r_k\}$, gdzie $r_i = (x^i_r, y^i_r)$, oznacza pole, w którym znajduje się punkt do którego platforma ma dostarczyć i-ty surowiec.
6) p - liczba platform
7) T - czas trwania jednego cyklu, czyli okresu w którym wszystkie surowce mają być dostarczone z punktu odbioru do punktu docelowego (jako jednostkę przyjmujemy czas, jaki platforma potrzebuje na przemieszczenie się do pola sąsiedniego - zakładamy, że dla każdej platformy czas ten jest taki sam.

Zmiennymi decyzyjnymi są ciągi $h_1 = (h_{11}, h_{12}, \ldots), h_2, \ldots, h_p$, reprezentujące przypisanie surowców do platform w określonej kolejności, czyli są to ciągi składające się z liczb 1,2,..., k, przy czym każda liczba będzie występowała łącznie we wszystkich ciągach $h_1, h_2, \ldots, h_p$ dokładnie raz . Zdefiniujmy również:

$H = h_1, h_2, \ldots, h_p$

Ograniczenia - Przy zdefiniowanej przestrzeni zmiennych decyzyjnych nie ma ograniczeń zadania optymalizacyjnego.

Funkcją celu jest funkcja $N(H)$, przy czym sposób jej obliczania jest umieszczony poniżej.

Aby obliczyć $N(H)$ należy rozwiązać następujące zadanie optymalizacyjne:

Przestrzeń zmiennych decyzyjnych: ciągi ciągów par liczb całkowitych o długości T - $(a_1, a_2, \ldots a_p)$ - gdzie $a_i = (a_{i1}, a_{i2}, \ldots a_{iT})$. (liczby a_{ij} oznaczają pola na których kolejno znajdują się platformy). Jako A oznaczmy zbiór wszystkich ciągów $(a_1, a_2, \ldots a_p)$ spełniających poniższe ograniczenia (jest to zatem zbiór decyzji dopuszczalnych):

Ograniczenia:

1) Dla danych $i \in \{1, 2, \ldots p\}$, $j \in \{1, 2, \ldots T-1\}$ liczby a_{ij}, $a_{i(j+1)}$ są od siebie różne - i wtedy oznaczają sąsiednie pola, albo $a_{ij} = a_{i(j+1)} = a_{i(j+2)} = a_{iT}$
2) Dla danego $j \in \{1, 2, \ldots T\}$ $a_{uj} \neq a_{vj}$ dla dowolnych $u, v \in \{1, 2, \ldots p\}$, przy czym u jest różne od v - ten warunek zapewnia, że dwie platformy nie znajdą się na tym samym polu równocześnie
3) Dla danego $j \in \{1, 2, \ldots T-1\}$ $a_{uj} = a_{v(j+1)} \Rightarrow a_{u(j+1)} \neq a_{vj}$ dla dowolnych $u, v \in \{1, 2, \ldots p\}$, przy czym u jest różne od v- ten warunek zapewnia, że platformy nie zamienią się miejscami (co powodowałoby ich zderzenie).
4) Dla wszystkich $i \in \{1, 2, \ldots p\}$, $j \in \{1, 2, \ldots T\}$ $a_{ij} \notin F$
5) $a_{i1} = q_{hi1}$, dla $i \in \{1, 2, \ldots p\}$
6) $a_{iT} = q_{hi1}$, dla $i \in \{1, 2, \ldots p\}$
7) W ciągu a_i liczby $q_{hi1}, r_{hi1}, q_{hi2}, r_{hi2}, \ldots$ muszą wystąpić w podanej kolejności (co nie znaczy, że nie mogą występować też w innych miejscach ciągu w innej kolejności lub w innych ciągach).

Funkcja celu, która ma zostać zminimalizowana:

$f(a_1, .., a_n) = \sum_{i=1}^{n} \sum_{j=1}^{T-1} d(a_{i,j}, a_{i,j+1})$ gdzie $d(a,b)$ oznacza odległość między punktami a, b w metryce miejskiej - mówiąc obrazowo oznacza to łączną długość dróg przebytą przez platformy.

Funkcja $N(H)$ jest zdefiniowana następująco:

$N(H) = \min_{A} f(a_1, .., a_n)$, jeżeli rozwiązanie powyższego zadania optymalizacji istnieje,

$N(H) = p \cdot (T+1)$, jeżeli nie istnieje.

d) Zastosowania

Projekt może mieć liczne zastosowania przy planowaniu sposobu rozwożenia surowców np. na hali produkcyjnej przez autonomiczne platformy (właśnie takim problemem był zainspirowany ów projekt). Z pewnymi modyfikacjami podobne algorytmy mogłyby także być zastosowane do:

- planowania przemieszczania się autobusów rozwożących ludzi na dużych lotniskach
- ogólnie przewożenia materiałów w ramach fabryk
- W wersji "trójwymiarowej" - planowania ruchu samolotów tak, aby się nie zderzały
- Planowania pracy kurierów i listonoszy

3. Opis algorytmu

a) Idea

Zadanie polega na rozwiązaniu problemu optymalizacyjnego w którym zagnieżdżony jest drugi problem optymalizacyjny, który należy rozwiązać w celu wyliczenia funkcji celu. Problem zagnieżdżony, czyli znalezienie optymalnych tras platform przy ustalonym przyporządkowaniu surowców do platform zostanie rozwiązany w sposób deterministyczny, za pomocą modyfikacji algorytmu A*, natomiast zewnętrzny problem, czyli przyporządkowanie surowców do platform wraz z kolejnością ich obsługiwania zostanie rozwiązany przy pomocy algorytmu genetycznego.

b) Schemat algorytmu A*

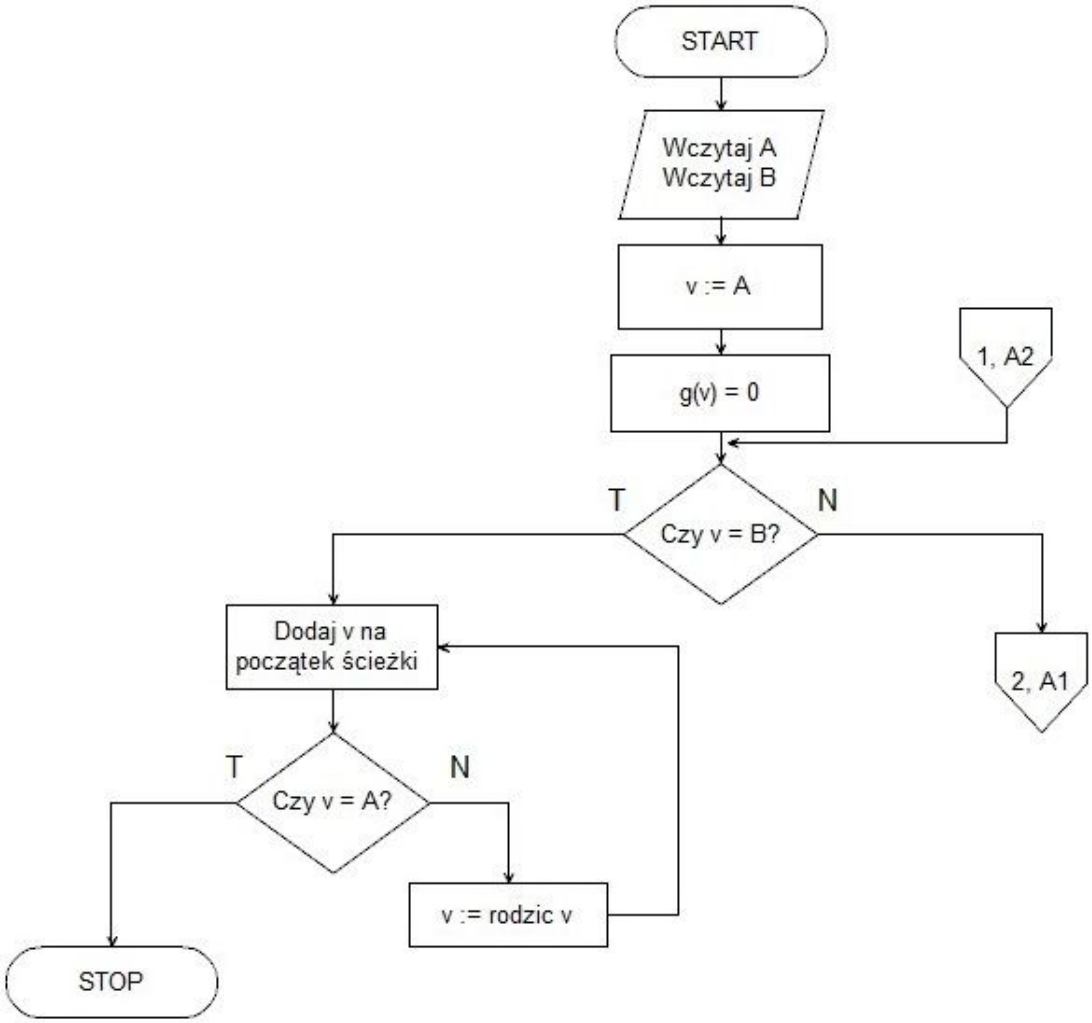
Algorytm A* służy do wyszukiwania najkrótszej drogi między dwoma wierzchołkami w grafie. Dla każdego wierzchołka v grafu są wykorzystywane funkcje:

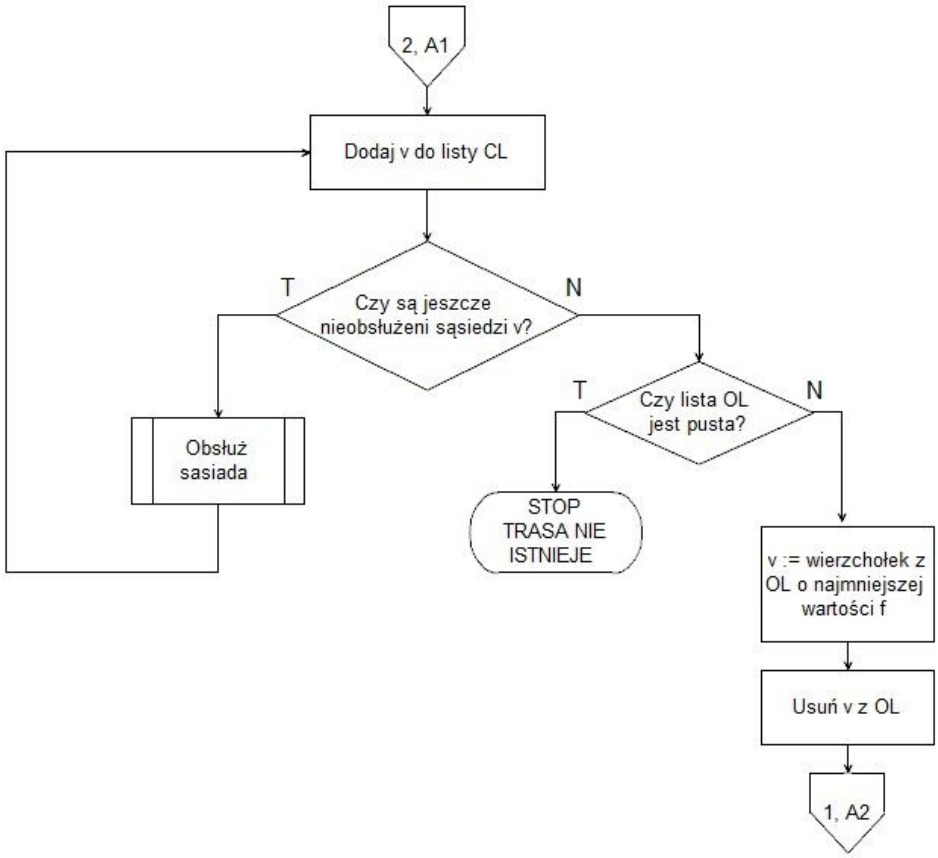
$g(v)$ - odległość wierzchołka od punktu początkowego

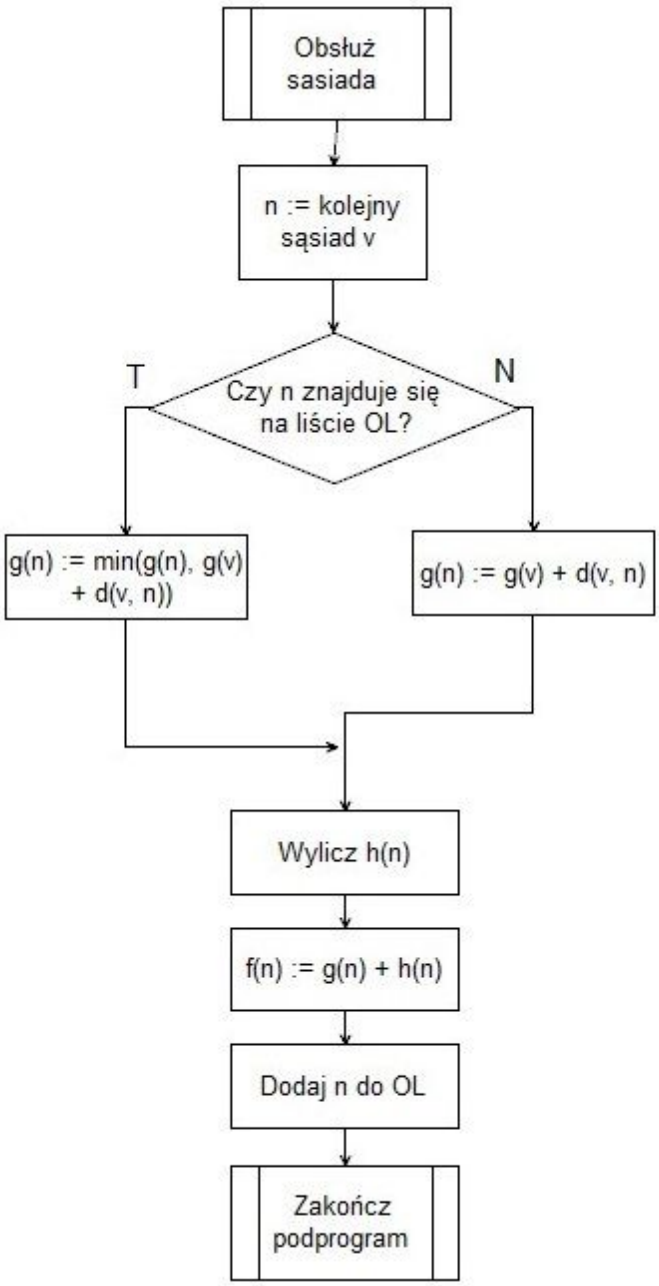
$h(v)$ - przewidywana przez heurystykę odległość v od wierzchołka końcowego

$f(v) = g(v) + h(v)$

Działanie podstawowej wersji algorytmu przedstawiono na poniższym schemacie blokowym (lista CL - lista wierzchołków zamkniętych, OL - otwartych, $d(v_1, v_2)$ - długość krawędzi między wierzchołkami (v_1, v_2)):


START
Wczytaj A
Wczytaj B
v := A
g(v) = 0
1, A2
T
Czy v = B?
N
Dodaj v na początek ścieżki
2, A1
T
Czy v = A?
N
v := rodzic v
STOP


2, A1
Dodaj v do listy CL
T
Czy są jeszcze nieobsłużeni sąsiedzi v?
N
Obsłuż sasiada
T
Czy lista OL jest pusta?
N
STOP TRASA NIE ISTNIEJE
v := wierzchołek z OL o najmniejszej wartości f
Usuń v z OL
1, A2



c) Schemat algorytmu genetycznego

Poniżej umieszczono schemat blokowy podstawowego algorytmu genetycznego:

Wymagane jest tu objaśnienie pewnych elementów schematu. Algorytmy genetyczne bazują na *populacji* rozwiązań traktowanych jako *osobniki* populacji - pojedyncze rozwiązanie jest zapisane za pomocą *genotypu*, na który składają się *chromosomy*, czyli uporządkowane ciągi tzw. *genów*. W szczególności genotyp może stanowić pojedynczy chromosom.

Jakość każdego osobnika w kontekście rozwiązania problemu jest oceniana za pomocą *funkcji dopasowania*. Na podstawie tej funkcji dokonywana jest *selekcja* osobników, czyli wybór podzbioru populacji, który będzie uczestniczył w tworzeniu nowego pokolenia (czyli nowej populacji). Często wykorzystywane jest skalowanie funkcji celu (czyli zmiana jej wartości według ustalonego schematu, najczęściej wykorzystująca rozkład oceny rozwiązań w populacji) mające na celu zabezpieczenie przed utknięciem algorytmu w optimum

lokalnym. Z tego też względu wykorzystywane są różne metody selekcji (m. in selekcja metodą koła ruletki, selekcja turniejowa).

Osobniki wyłonione przez selekcję są poddawane działaniu *operatorów genetycznych* - *mutacji* oraz *krzyżowaniu*. Do mutacji wykorzystywany jest pojedynczy osobnik i polega ona na zmianie jego pojedynczych genów, w wyniku czego powstaje nowy osobnik. Krzyżowanie wykorzystuje natomiast wybrane losowo pary osobników - poprzez łączenie ich genotypów (odpowiedni dobór genów z jednego i drugiego osobnika) tworzony jest nowy osobnik. Krzyżowaniu powinna podlegać znacznie większa liczba osobników. W wyniku zastosowania tych operatorów tworzona jest nowa populacja rozwiązań.

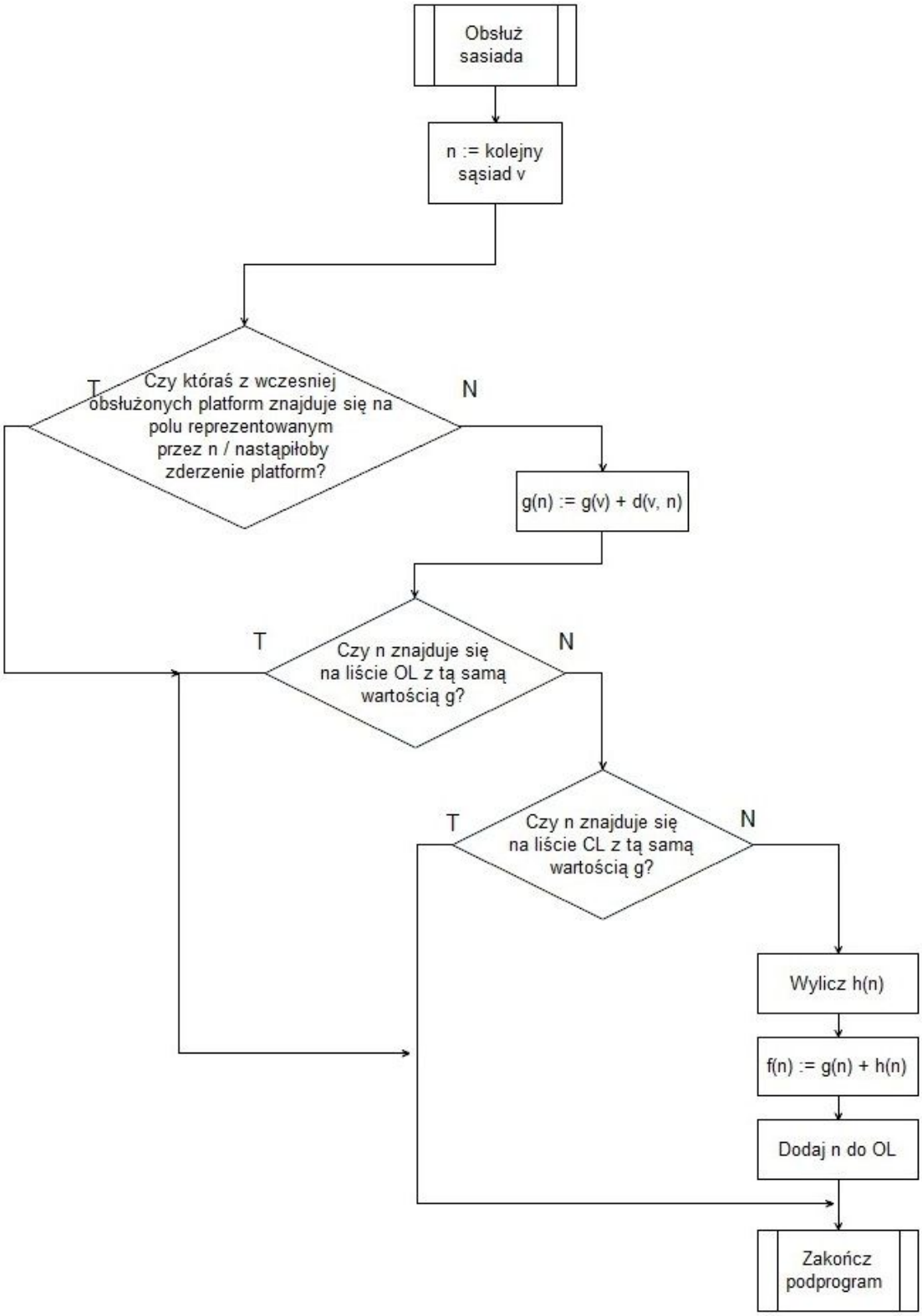
Najczęściej stosowanym i najbezpieczniejszym kryterium jest maksymalna liczba generacji, czyli iteracji algorytmu. Inne stosowane kryteria wykorzystują np. entropię zbioru rozwiązań.

d) Adaptacja - algorytm A*

W przypadku, gdy szukamy najkrótszej drogi na planszy o kwadratowych polach, wierzchołkami grafu są poszczególne pola, natomiast krawędzie istnieją między sąsiednimi polami i ich długość jest równa 1. Dodatkowo część pól jest zabronionych, przez co odpowiadające im wierzchołki są usuwane z grafu. Wykorzystany algorytm był bardziej inspirowany algorytmem A* gdyż liczba modyfikacji jest bardzo duża. Po pierwsze, algorytm poszukiwania drogi między dwoma punktami jest wykonywany wielokrotnie dla każdej platformy, według schematu:

punkt poboru 1 surowca -> punkt docelowy 1 surowca -> punkt poboru 2 surowca -> punkt docelowy 2 surowca -> … punkt poboru ostatniego surowca -> punkt ostatniego surowca -> punkt poboru 1 surowca

Dodatkowo, w rzeczywistości pola zabronione są dla poszczególnych platform są różne w zależności od chwili czasu, gdyż w danej chwili na pewnym polu może się znajdować inna platforma, a także nie może dojść do sytuacji, w której platforma zamieniałaby się miejscem z inną w trakcie jednego ruchu (gdyż wtedy nastąpiłoby zderzenie). W związku z tym w zależności od chwili czasu chwilowo są usuwani niektórzy sąsiedzi danego wierzchołka. Dodatkowo, w tym przypadku może dojść do sytuacji, gdy platforma będzie musiała przejść przez jakieś pole więcej niż raz. W celu wprowadzenia tych zmian wykorzystano taki schemat blokowy, jak dla algorytmu A*, jedynie ze zmienionym podprogramem obsłuż sąsiada, który jest zaprezentowany poniżej:



Taki algorytm zapewnia zarówno brak kolizji, możliwość kilkukrotnego przejścia przez dane pole, a także nie jest możliwe, że nastąpi jego zapętlenie.

e) Adaptacja - algorytm genetyczny

i) Kodowanie chromosomu

W zastosowanym przez nas algorytmie genotyp jest pojedynczym chromosomem. Musi on zawierać informację na temat przyporządkowania surowców do platform i kolejności ich obsługiwania przez dane platformy.

Chromosom to macierz $C = [c_{ij}]$, gdzie i oznacza numer platformy, j - numer surowca. Jeżeli platforma o numerze i nie jest przyporządkowana do surowca o numerze j, to $c_{ij} = 0$, natomiast jeśli jest, c_{ij} oznacza jako który z kolei jest obsługiwany przez j-tą platformę i-ty surowiec. Z tego opisu wynika, że w każdej kolumnie macierzy jest dokładnie jeden element niezerowy.

ii) Funkcja dopasowania

Funkcja dopasowania danego osobnika jest funkcja $N(H)$ dla danego przyporządkowania surowców do platform (funkcję dla chromosomu C odpowiadającemu przyporządkowaniu H również będziemy oznaczać przez N). Dodatkowo jest on skalowana:

$$f(C) = a \cdot N(C) + b$$

Gdzie współczynniki a oraz b są dobrane w taki sposób, aby dla $N(C)$ równemu wartości średniej $N(C)$ dla osobników z danej populacji $f(C)$ było równe 1, natomiast dla najlepszego rozwiązania była ona równa ustalonej liczbie r - podawanej jako parametr algorytmu. W wyniku tego najlepsze osobniki (a więc te o najmniejszej wartości funkcji $N(C)$) mają największą wartość funkcji $f(C)$. Podane skalowanie nie było konieczne w przypadku używania selekcji turniejowej (a taką ostatecznie zastosowane), jednak funkcja dokonująca skalowania była pisana wykonywana przed ustalaniem sposobu selekcji - a jej obecność umożliwia zastosowanie większej liczby sposobów.

iii) Metoda selekcji

Jako metodę selekcji wybrano metodę selekcji turniejowej. Z populacji jest wybierana bez zwracania grupa osobników o określonej liczności (jeden z podawanych parametrów algorytmu), po czym wybierany jest z niej osobnik o najwyższej wartości funkcji dopasowania. Dodatkowo w aplikacji algorytm przyjmuje parametr określający ile osobników najlepszych w populacji jest pewnych trafienia do wyselekcjonowanej grupy). W momencie jeśli pozostała liczba osobników nie wystarczyłaby do utworzenia liczby osobników która ma pozostać w wyniku selekcji, losowanie grup turniejowych jest wstrzymywana i z pozostających osobników dobierana jest taka ich liczba o najlepszym przystosowaniu, żeby uzupełnić grupę osobników która ma powstać wyniku selekcji.

iv) Mutacja

Zarówno w mutacji jak i krzyżowaniu wykorzystywana jest funkcja zapewnij_poprawność(C) mająca zapewnić, że gdy w każdej kolumnie chromosomu jest dokładnie jedna liczba niezerowa, liczby w wierszu są równe. Działa ona w ten sposób, że gdy w pewnym wierszu występuje kilka liczb o tej samej wartości to zapewniane jest, że liczby występujące w tych kolumnach są mniejsze od liczb, od których były mniejsze przed przekształceniem oraz większe od liczb, od których były większe przed przekształceniem natomiast kolejność liczb w obrębie tych kolumn jest dobierana losowo. Funkcja ta zapewnia również, że w każdym wierszu oprócz zer z przedziału $1, \ldots, s$ (gdzie s to liczba surowców obsługiwanych przez daną platformę) występują wszystkie liczby . Szczegółowy sposób implementacji tej funkcji nie zostanie tu opisany.

Sposób dokonywania mutacji prezentowany jest przez poniższy fragment pseudokodu - rozmiar_mutacji - dodatkowy parametr algorytmu (dla chromosomu - macierzy C, k - liczba platform, p liczba surowców):

```
u := rozmiar_mutacji
jeżeli rozmiar_mutacji < 0
        jeżeli k > 1 to
                u :=  losuj liczbę z przedziału od 1 do  [k/2]
        w przeciwnym wypadku
                k := 1
wybierz losowo u kolumn macierzy C
dla każdej z wybranych kolumn
    losuj wiersz w
    przepisz niezerowy element kolumny do wiersza w (i wyzeruj jego stare miejsce)
zapewnij_poprawność(C)
```

v) Krzyżowanie

Sposób dokonywania krzyżowania chromosomów $C1$, $C2$ przedstawiono w poniższym pseudokodzie (k - liczba platform, p liczba surowców):

```
Wybierz losowo u kolumn (u <= k) z chromosomu C1
Dobierz pozostałe kolumny z chromosomu C2
W kolumnach  z C2 niezerowe elementy zwiększ o k
Utworz chromosom C3 z wybranych kolumn
zapewnij_poprawność(C3)
```

W ten sposób fragmenty tras z jednego chromosomu są wstawiane do drugiego.

vi) Nowa populacja

Nowa populacja w algorytmie tworzona jest przez połączenie zbioru osobników wybranych w procesie selekcji,

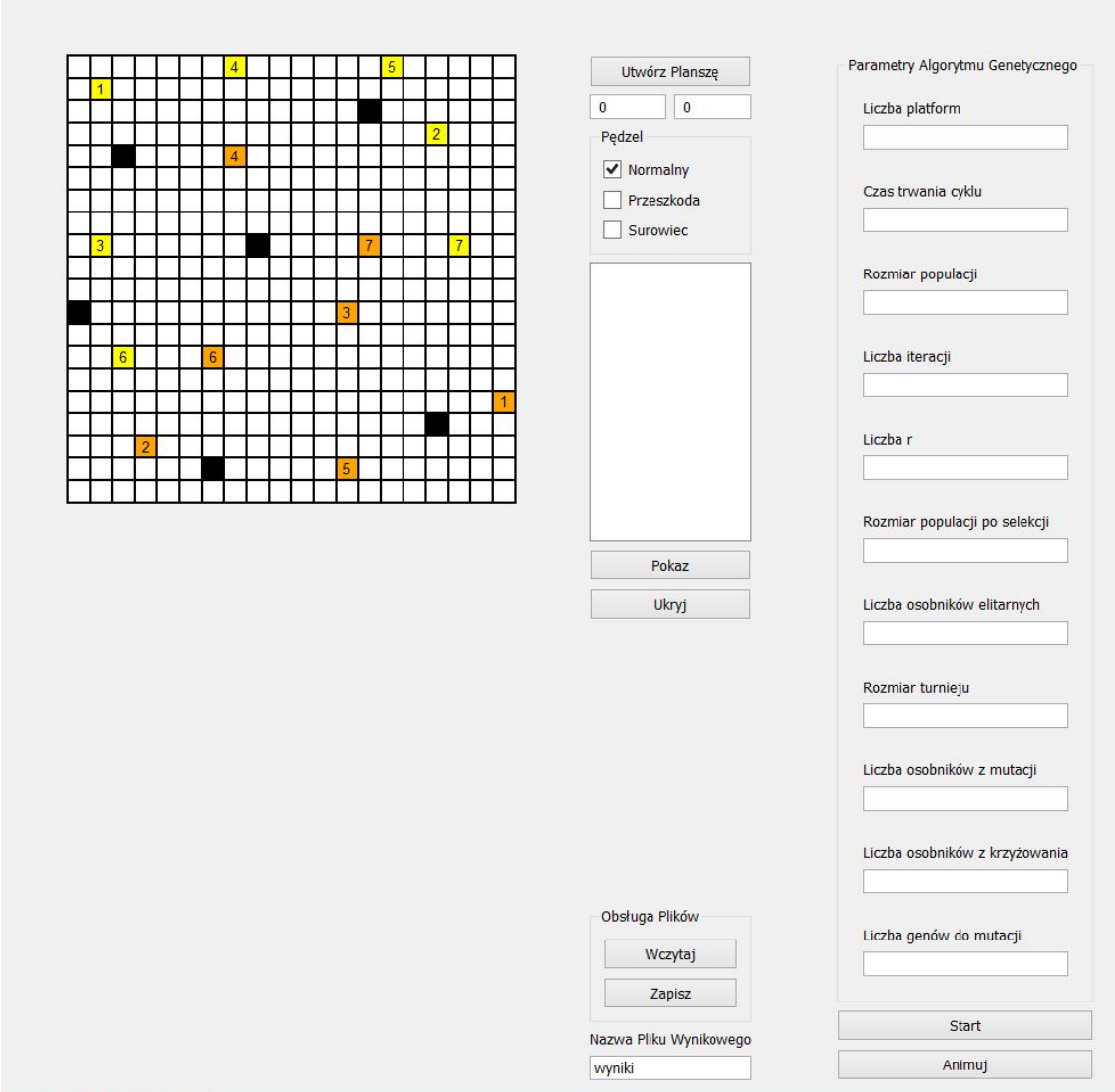
4. Aplikacja

W celu zapewnienia odpowiedniego komfortu pracy została wykonana aplikacja okienkowa wraz z interfejsem graficznym. Docelowo aplikacja zapewniać miała spełniać następujące warunki:

- możliwość graficznego tworzenia instancji problemu o zadanych wymiarach
- możliwość graficznej edycji danej instancji problemu
- możliwość zapisu danej instancji do pliku
- możliwość modyfikowania parametrów algorytmu genetycznego
- animację wynikowego ruchu platform
- wizualizację wygenerowanych ścieżek - element ten nie jest wykorzystywany w wersji finalnej, był on jednak konieczny na etapie implementacji algorytmu znajdowania najkrótszej drogi

Postawione cele zostały zrealizowane przy wykorzystaniu popularnej biblioteki Qt4 w wejrsji dla w języka Python służącej do tworzenia aplikacji okienkowych. Udostępnienia ona wygodne narzędzia do realizacji tego celu, takie jak funkcje rysujące linie czy koła

Poniżej przedstawiono przykładowy zrzut ekranu z działania aplikacji:



Tworzenie nowej planszy reprezentującej dany problem odbywa się za pośrednictwem przycisku **Utwórz Planszę** ulokowanego w centralnej części okna, oraz dwóch pól edycyjnych ulokowanych pod nim służących do zadawania rozmiarów planszy.

Po wciśnięciu w.w. przycisku z lewej strony okna pojawia się pusta siatka o wskazanych wymiarach. Jej edycja odbywa się za pomocą kursora myszy. Kliknięcie na danej komórce siatki powoduje zmianę jej typu na wskazany przez obecnie aktywne pole wyboru znajdujące się w ramce z tytułem **Pędzel**.

W zależności od wyboru pędzla, zachowanie się danego pola siatki jest następujące:

- pędzel **Normalny -** dane pole ustawione jest na kolor **biały -** jest ono interpretowane przez algorytm jako pole dopuszczone do ruchu - nie znajduje się w nim ani przeszkoda, ani pole odbioru surowca - domyślnie w tym stanie znajdują się wszystkie pola siatki po jej utworzeniu. Należy zwrócić uwagę, że o ile ustawianie w tym stanie pól figurujących już jako przeszkody odbywa się swobodnie, o tyle pola reprezentujące **pola poboru i odbioru danego surowca** można ustawiać jako

normalne (kasować) **tylko w kolejności przeciwnej do kolejności dodawania** - ma to związek z zachowaniem numeracji już dodanych pól

- pędzel **Przeszkoda** **-** dane pole ustawione jest na kolor **czarny** - jest ono interpretowane jako pole nie dopuszczone do ruchu platform

- pędzel **Surowiec** **-** pozwala na ustawianie danej komórki, która znajduje się w stanie **Normalnym** jako punkt poboru lub odbioru danego surowca.
    - Jeżeli żadne pole nie jest ustawione jako pole surowca, kliknięta komórka przybiera kolor **żółty** z numerem **1** co jest interpretowane jako punkt **z którego pobierany jest surowiec** o numerze 1
    - Jeżeli **poprzednio** dodano pole **poboru** surowca, dany punkt interpretowany jest jako pole **odbioru** surowca o **tym samym numerze surowca co pole ustawione poprzednio** i przybiera ono kolor **pomarańczowy**
    - Jeżeli **poprzednio** dodano pole **odbioru** surowca, dany punkt interpretowany jest jako pole **poboru** surowca o **numerze surowca większym o jeden niż pole ustawione poprzednio** i przybiera ono kolor **żółty**

Znajdująca się pod polem wyboru pędzli lista oraz przyciski **Pokaż** i **Ukryj** służą do analizy wygenerowanych ścieżek i nie znajdują zastosowania w gotowej aplikacji. Ścieżkę można dodać do listy poprzez metodę *mainWindow.grid.addPath([[x1, y1], [x2, y2], ...])* (jako argument przyjmuje ona listę kolejnych punktów zapisanych w postaci dwuelementowych list.

Przyciski **Wczytaj** i **Zapisz** znajdujące się w ramce zatytułowanej **Obsługa Plików** służą do obsługi pliku przechowującego daną instancję planszy. Po naciśnięciu każdego z nich wywoływane jest okno dialogowe umożliwiające wybór miejsca odczytu/zapisu planszy.

Pole tekstowe "**Nazwa Pliku Wynikowego**" - podczas wykonywania algorytmu genetycznego w pliku o tej ścieżce dostępu będą zapisywane w kolejnych liniach wyniki kolejnych iteracji w formacie numer_iteracji;łączna_suma_długości_dróg

Pola tekstowe w dziale "**Parametry algorytmu genetycznego**" :

Liczba platform - maksymalna liczba platform przemieszczających się po planszy (może się zdarzyć, że do niektórych platform nie będzie przyporządkowany żaden surowiec i wtedy są one usuwane z planszy).

Czas trwania cyklu - czas w którym wszystkie surowce mają być dostarczone do swoich punktów docelowych.

Rozmiar populacji - liczba osobników w pierwszej populacji w algorytmie genetycznym

Liczba iteracji - liczba generacji algorytmu genetycznego.

Liczba r - liczba r wspomniana w punkcie 3 sprawozdania podpunkt ii) - w rzeczywistości nie ma znaczenia, może być to dowolna liczba dodatnia.

Rozmiar populacji po selekcji - liczba osobników, która zostanie wykorzystana w procesie tworzenia nowej populacji, zostaną one wybrane w wyniku selekcji.

Liczba osobników elitarnych - liczba osobników o najlepszej ocenie przystosowania, które na pewno zostaną wybrane w wyniku selekcji

Rozmiar turnieju - liczba osobników branych do każdego turnieju w procesie selekcji (patrz punkt 3 podpunkt iii)

Liczba osobników z mutacji - liczba osobników w nowej populacji, które powstaną w wyniku mutacji

Liczba osobników z krzyżowania - liczba osobników w nowej populacji, które powstaną w wyniku krzyżowania

Liczba genów do mutacji - w chromosomie (będącym macierzą) podczas mutacji jest modyfikowana liczba kolumn dana tą wartością. w przypadku wpisania wartości ujemnej liczba ta wybierana jest losowo.

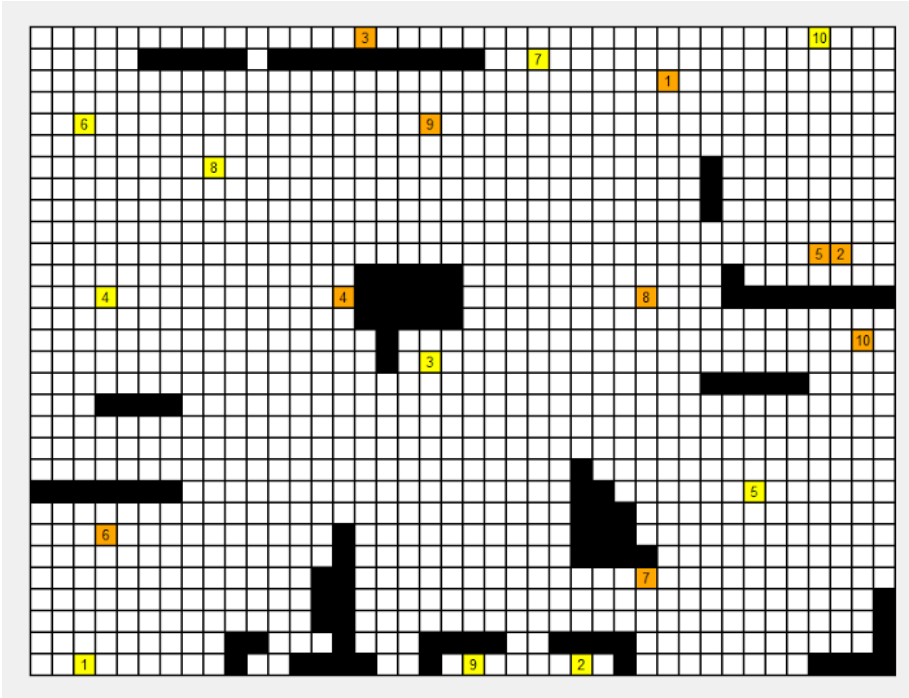
Przycisk Start - powoduje uruchomienie algorytmu genetycznego

Przycisk Animuj - włącza animację obrazującą ruch platform po planszy - po wykonaniu algorytmu genetycznego.

5. Testy

Ze względu na specyfikę problemu, przede wszystkim konieczność spełnienia warunku aby **rozmiar populacji = rozmiar populacji po selekcji + liczba osobników z krzyżowania + liczba osobników z mutacji** (zapewnia on że w każdej iteracji algorytmu liczba osobników jest stała, inaczej dochodzi do jej zmiany po pierwszej iteracji) nie dało się zestawić wyników testów w zbiorczych tabelach obrazujących wpływ jednego parametru, gdyż zmiana jednego z parametrów w większości pociągała zmianę pozostałych.

Poniżej przedstawiono 10 wygenerowanych przypadków testowych, które pozwalają porównać wpływ poszczególnych parametrów na wartość funkcji celu w kolejnych iteracjach dla poniższej instancji testowej o wymiarach **30X40** dla **100 iteracji**



Zestawiając ze sobą wyniki dla przebiegów (1 2 3) z (4 5 6) jesteśmy w stanie wnioskować o wpływie **liczby osobników elitarnych (2 i 0).**

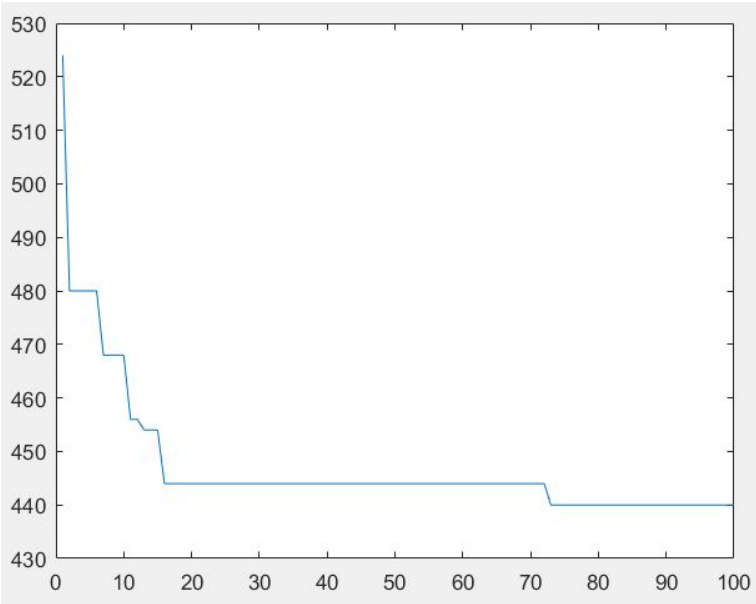
Zestawiając wyniki (1 z 7) , (2 z 8), (3 z 9) można wnioskować o wpływie **stosunku liczby osobników z mutacji do liczby osobników z krzyżowania.**

Ze względu na długi czas wykonywania się programu (np. dla problemu porównywalnym z 2 na komputerze wyposażonym w dwurdzeniowy procesor Intel Core i7 5500U o maksymalnej częstotliwości taktowania 3GHz oraz 16GB pamięci RAM osiągano czas rzędu 10 minut) pominięto mniej znaczące parametry jak np. liczbę genów do mutacji,

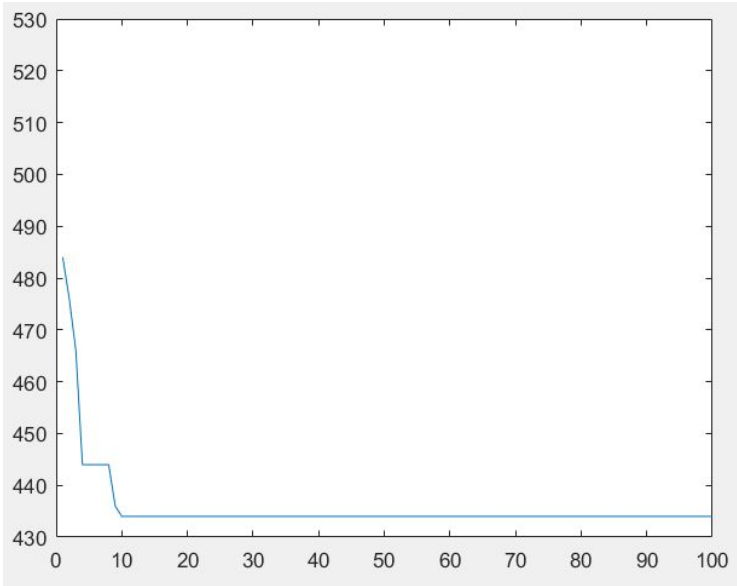
która w klasycznym algorytmie genetycznym zwykle przyjmuje się równą 1 (w odniesieniu do sytuacji w przyrodzie mutacje są najczęściej drobne)

Parametr	Rozmiar populacji	Rozmiar populacji po selekcji	Liczba osobników elitarnych	Rozmiar turnieju	Liczba osobników z mutacji	Liczba osobników z krzyżowania	Liczba genów do mutacji	Najlepszy wynik
L. p.								
1	20	10	2	2	3	7	1	440
2	40	10	2	2	5	25	1	434
3	10	5	2	2	1	4	1	456
4	20	10	0	2	3	7	1	434
5	40	10	0	2	5	25	1	434
6	10	5	0	2	1	4	1	434
7	20	10	2	2	2	8	1	434
8	40	10	2	2	10	20	1	434
9	10	5	2	2	2	3	1	434
10	20	10	2	2	8	2	1	434

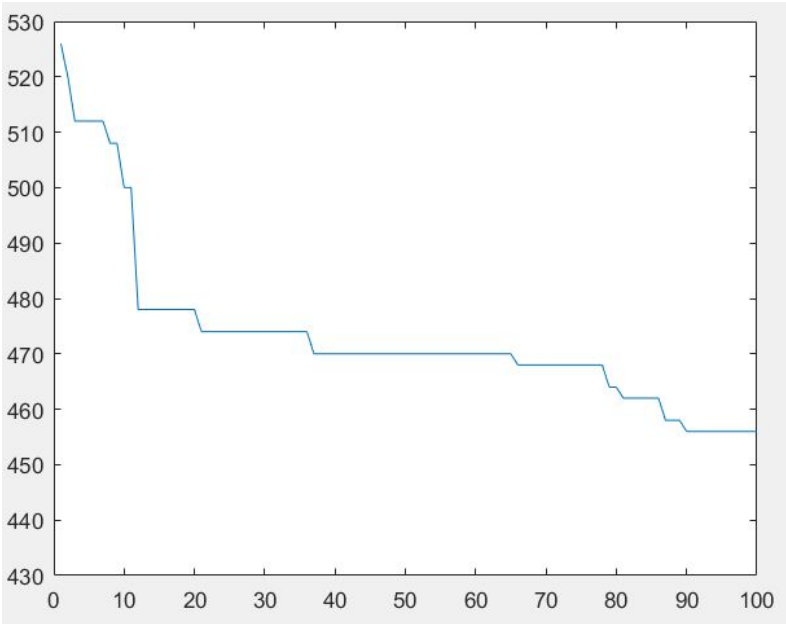
Wartości parametrów dla poszczególnych wyników



Wyniki 1

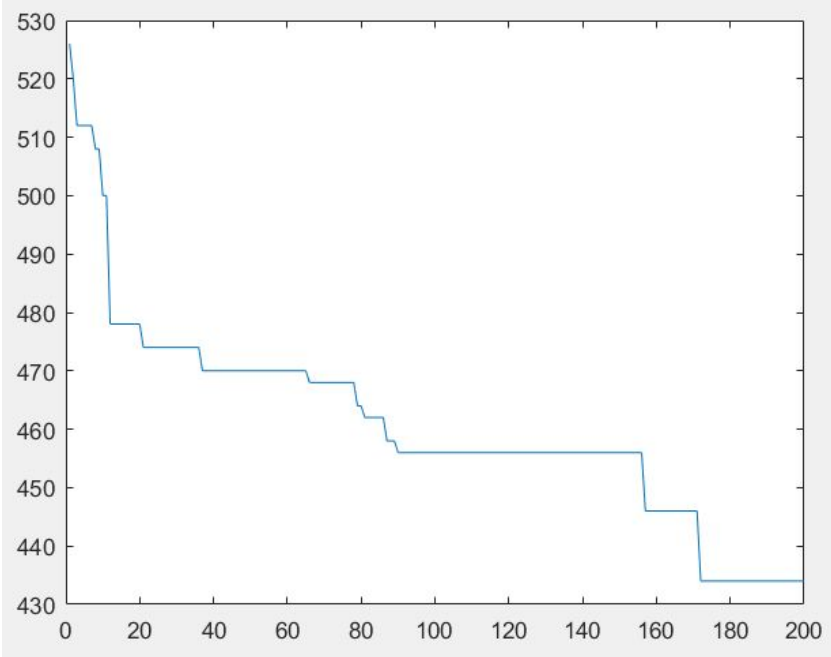


Wyniki 2

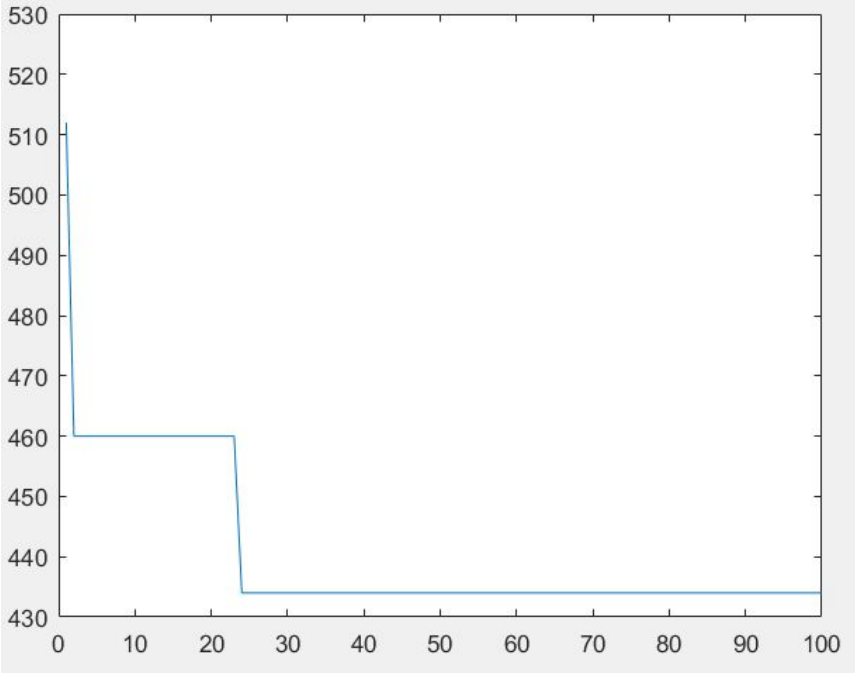


Wyniki 3

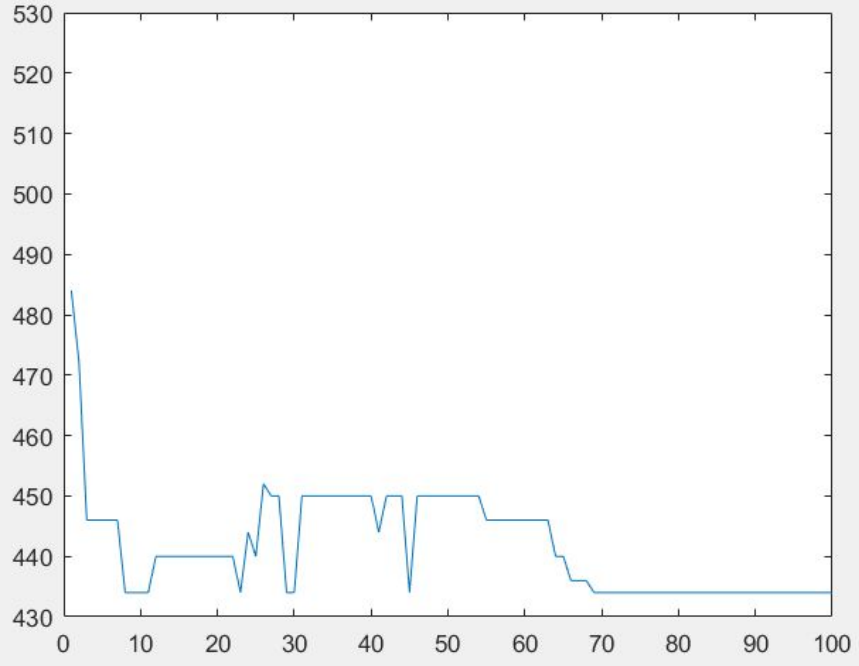
Ponieważ dla powyższego wykresu widać, że ostateczny wynik jest gorszy dla podanych parametrów niż dla innych parametrów, a do tego jeszcze pod koniec 100 iteracji dosyć często następowała zmiana, więc sprawdzono działanie dla 200 iteracji. Osiągnięto wartość 434 a wykres wyniku w zależności od numeru iteracji jest widoczny poniżej:



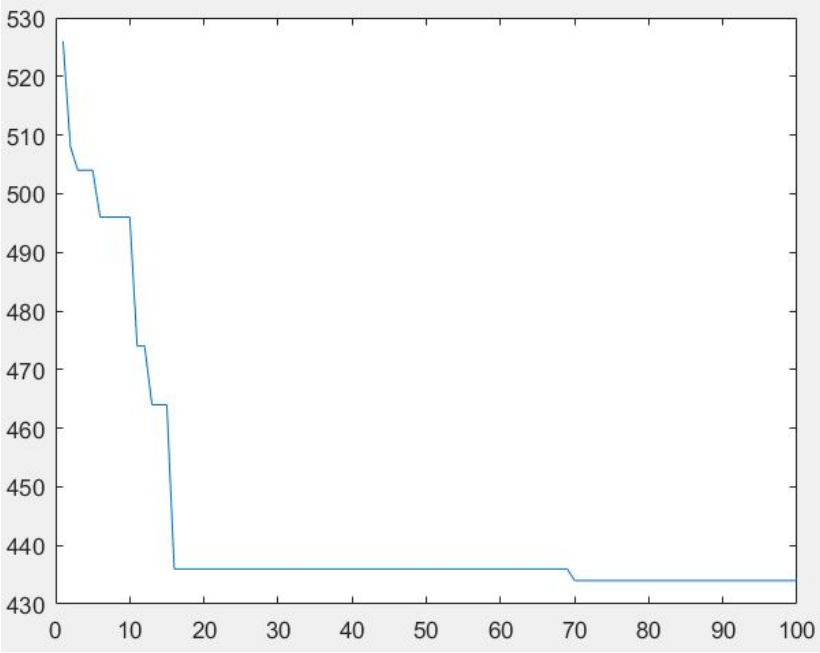
Wyniki 3 dla 200 iteracji



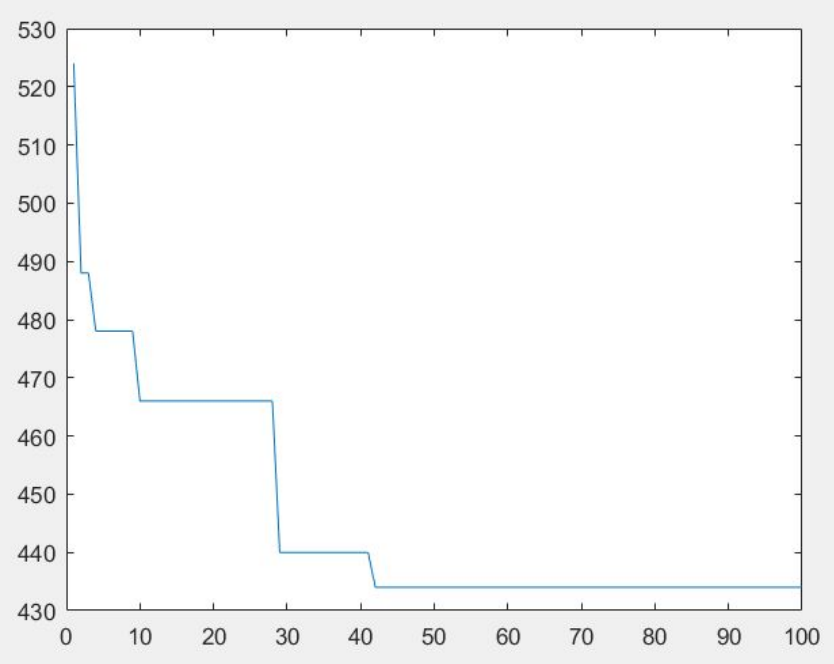
Wyniki 4



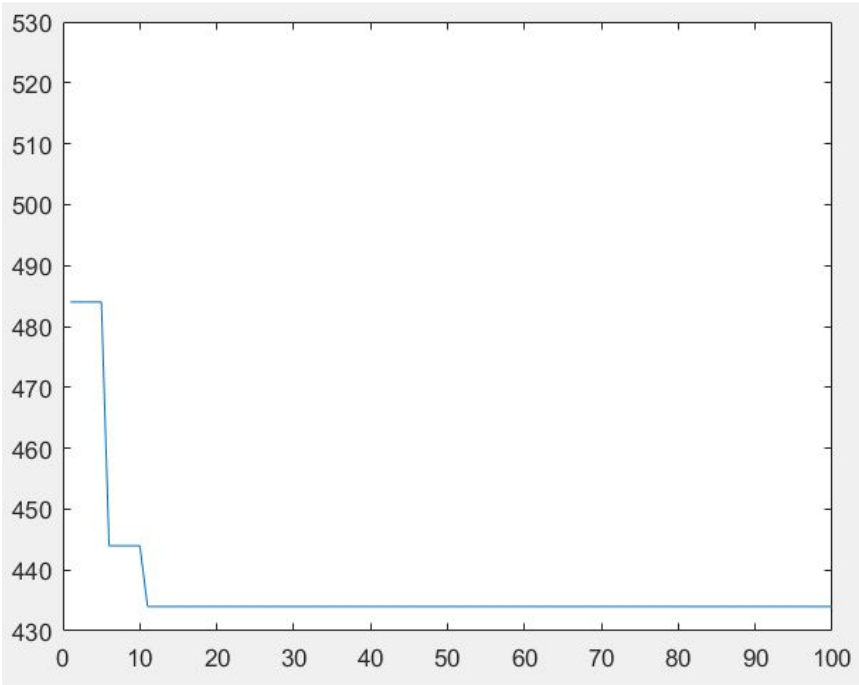
Wyniki 5



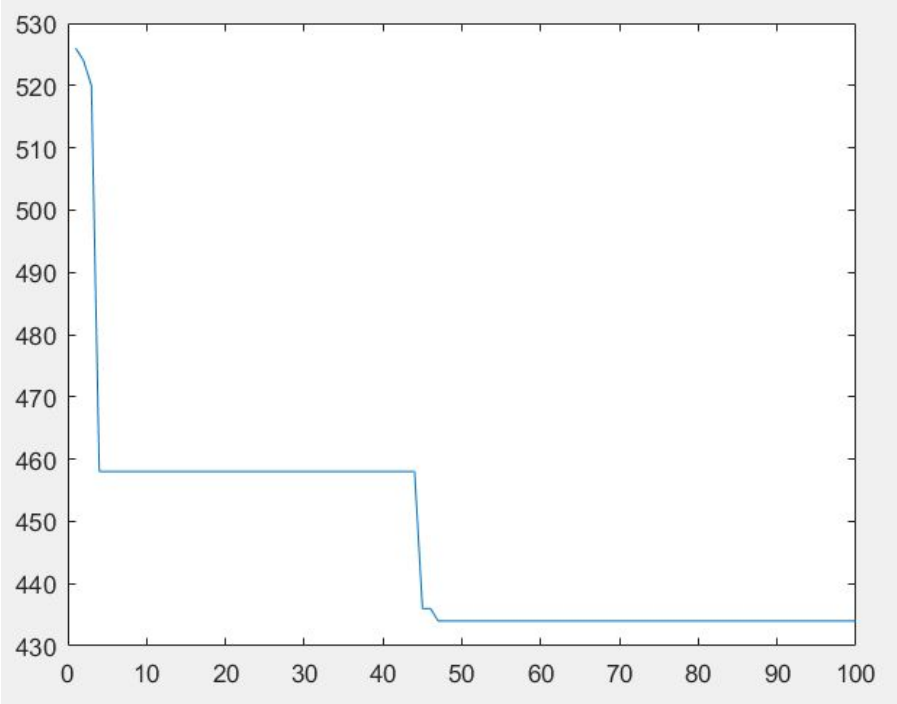
Wyniki 6



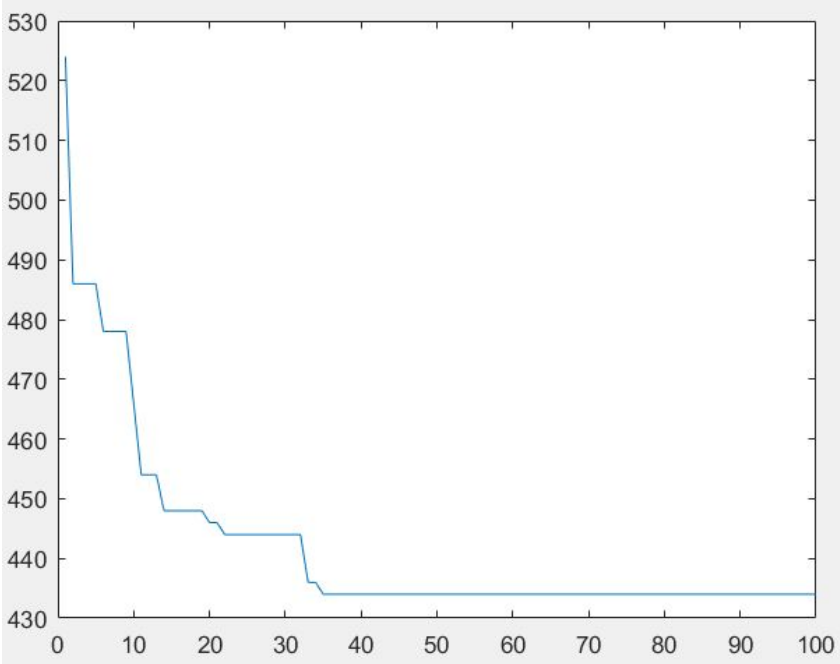
Wyniki 7



Wyniki 8



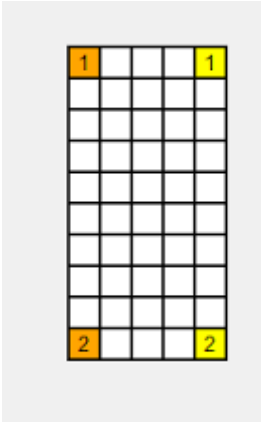
Wyniki 9



Wyniki 10

6. Rozwiązania dla wybranych przypadków

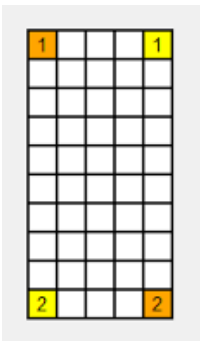
a. Prosty przypadek 10x5 - p. odbioru na jednym boku



- optymalne rozwiązanie dla 1 platformy ma koszt = [(5-1)]+ [(10-1) + (5-1)] + [(5-1)] + [(5-1) +(10-1)] = 34 (platforma wraca do punktu startowego)
  Rozwiązanie to jest generowane przez program po 1 iteracji

- optymalne rozwiązanie dla 2 platform ma koszt = [(5-1)+ (5-1)] + [(5-1) +(5-1)] = 16
  Rozwiązanie to jest generowane przez program po 1 iteracji
- Rozwiązania dla większej liczby platform są niedopuszczalne

b. Prosty przypadek 10x5 - p. odbioru w przeciwległych wierzchołkach

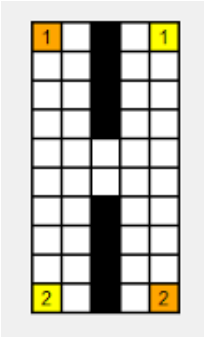


- Optymalne rozwiązanie dla 1 platformy ma koszt = [(5-1)]+ [(10-1)] +[(5-1)] + [(10-1)] = 26 (platforma wraca do punktu startowego)
- Rozwiązanie to osiągane jest przez algorytm po jednej iteracji

- Optymalne rozwiązanie dla 2 platform ma analogiczny koszt jak a przypadku a) wynoszący 16
- Rozwiązanie to osiągane jest przez algorytm po jednej iteracji

- Rozwiązania dla większej liczby platform są niedopuszczalne

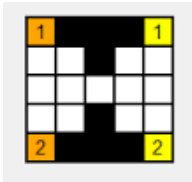
c. Prosty przypadek 10x5 - p. odbioru w przeciwległych wierzchołkach - przeszkody pomiędzy punktami dostawy i odbioru



- Optymalne rozwiązanie dla 1 platformy ma koszt -o 2*2*(5-1) = 16 większy niż w przypadku b) wynoszący 42
- Rozwiązanie generowane jest przez algorytm po 1 iteracji

- Optymalne rozwiązanie dla 2 platform ma koszt o 2*2*2*(5-1) = 32 większy niż w przypadku b) wynoszący 48, co sprawia, że optymalniejsze jest użycie jednej platformy - 42
- Rozwiązanie to (z jedną platformą) osiągane jest przez algorytm po jednej iteracji
- Rozwiązania dla większej liczby platform są niedopuszczalne

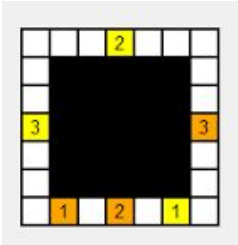
d. Prosty przypadek 5x5 - p. odbioru na jednym boku - wąski kanał



- optymalne rozwiązanie dla 1 platformy ma koszt 4*8 =32
- Rozwiązanie generowane jest przez algorytm po 1 iteracji

- Jest to również rozwiązanie optymalne dla dwóch platform, które program osiąga po jednej iteracji

e. Przypadek 7x7- surowce na obwodzie

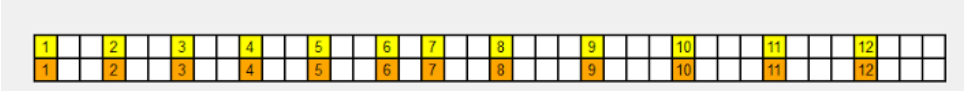


Wyniki dla jednej platformy:

- 1 - 2 - 3 - 1 - koszt 57
- **1 - 3 - 2 - 1 - koszt 40**
- **2 - 3 - 1 - 2 - koszt 40**
- **2 - 1 - 3 - 2 - koszt 40**
- 3 - 1 - 2 - 3 - koszt 48
- 3 - 2 - 1 - 3 - koszt 44

- W zależności od parametrów algorytmu rozwiązanie optymalnie o koszcie 40 osiągane jest po około 2 - 8 iteracjach

f. Przypadek 2x40 - znaczna liczba surowców



- Przypadek wygenerowany w celu sprawdzenia działania algorytmu genetycznego
- Rozwiązanie optymalne ma koszt 12*[(2-1) + (2-1)] = 24
- Dla parametrów: liczba platform = 12, czas trwania = 1000, rozmiar populacji =10, liczba iteracji = 200, rozmiar populacji po selekcji = 5, liczba osobników elitarnych = 2, rozmiar turnieju = 2, liczba osobników z mutacji = 1, liczba osobników z krzyżowania = 4, liczba genów do mutacji = 1 program osiąga to rozwiązanie w 91 iteracji

7. Wnioski

1) Przy p platformach i k surowcach łączna liczba możliwych przypadków przyporządkowania to:

$$k! \cdot \binom{p+k-1}{k}$$

(Najpierw ustalamy surowce w określonej kolejności, a później wstawiamy $p-1$ "przegród" między nimi, które to przegrody są miejscem podziału między platformami).

Dla przypadku testowego (10 platform, 10 surowców) daje to 335221286400 przypadków, co szacując czas obliczania najkrótszych dróg dla pojedynczego, skomplikowanego przypadku na 1s (a w rzeczywistości jest to zdecydowanie więcej jak wynika z obserwacji poczynionych podczas wywoływania algorytmu) oznacza, że sprawdzenie wszystkich przypadków zajęłoby w zaokrągleniu 10630 lat co potwierdza sensowność zastosowania algorytmów przybliżonych takich jak algorytm genetyczny.

2) Z uwagi na brak gotowych instancji testowych bardzo trudno było przetestować działanie algorytmu genetycznego - w podanych prostych przypadkach dla mniejszej liczby platform w rzeczywistości sprawdzane jest działanie zmodyfikowanego algorytmu A*, a jedynie dla przypadku z większą liczbą platform (ostatnia instancja) w rzeczywistości sprawdzono działanie algorytmu genetycznego. W każdym przypadku uzyskano poprawne wyniki.

3) Dla mniejszej populacji początkowej algorytm potrzebował na rozwiązanie problemu większej liczby iteracji, jednak uwzględniając znacznie krótszy czas jego wykonywania mniejsza liczność populacji może być bardziej opłacalna.

4) "Liczba osobników elitarnych" miała widoczne znaczenie jedynie, gdy rozmiar populacji po selekcji był ponad dwa razy mniejszy od rozmiaru populacji początkowej (co jest logiczne, gdyż w przeciwnym wypadku osobniki będące najlepsze i tak zostaną na pewno wyłonione w turniejach). Wtedy, przy braku strategii elitarnej algorytm bardzo szybko osiągał minimum, jednak dla późniejszych iteracji z niego wypadł. Może to mieć zalety (możliwość wydostania się z optimum lokalnego, które nie jest optimum globalnym) jak i wady (mimo, że osiągnięto wcześniej już optymalne rozwiązanie, później algorytm z niego wypada).

5) Poza wielkością populacji i liczbą osobników elitarnych inne parametry nie miały dającego się zaobserwować znaczenia - w związku z tym nie jest pewne, czy zastosowane w algorytmie metody krzyżowania faktycznie powodują "połączenie" dobrych cech osobników.

8. Literatura

[1] Michał Bereta, “Algorytmy genetyczne” - materiały do laboratorium “Zagadnienia Sztucznej Inteligencji:

[2] http://iair.mchtr.pw.edu.pl/~bputz/aisd_cpp/lekcja7/segment4/main.htm

[3] https://pl.wikipedia.org/wiki/Algorytm_A*