Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

PRACA MAGISTERSKA

MARTA RYŁKO, ANNA SKIBA

RÓWNOLEGŁE ALGORYTMY OPTYMALIZACJI TORU PRZEJAZDU W NARCIARSTWIE ALPEJSKIM

PROMOTOR:
dr inż. Roman Dębski

Kraków 2013

OŚWIADCZENIE AUTORA PRACY

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie, i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Computer Science, Electronics and Telecommunication

DEPARTMENT OF COMPUTER SCIENCE

MASTER OF SCIENCE THESIS

MARTA RYŁKO, ANNA SKIBA

PARALLEL ALGORITHMS FOR SKI-LINE OPTIMISATION IN ALPINE SKI RACING

SUPERVISOR:
Roman Dębski Ph.D

Krakow 2013

Serdecznie dziękuję …

Spis treści

1. Wstęp

Narciarstwo alpejskie to dyscyplina z długą historią. Rozwój sportowej wersji narciarstwa alpejskiego rozpoczął się w połowie XIX wieku, jednak nadal nie ma i prawdopodobnie nigdy nie będzie naukowej formuły opisującej tor po jakim należy się poruszać, aby zadaną trasę przejechać najszybciej. Ogromna ilość czynników, które wpływają na czas przejazdu znacznie utrudnia jej znalezienie. W sportowych dyscyplinach narciarstwa alpejskiego celem jest przejechanie w jak najkrótszym czasie wyznaczonej trasy od startu do mety, przejeżdżając przez wszystkie ustawione na trasie bramki - wymuszające skręty.

Problem, jakiego rozwiązania podejmujemy się w pracy, to problem optymalizacyjny rozwiązywany za pomocą symulacji komputerowej. Problem dotyczy znalezienia optymalnego toru przejazdu narciarza po trasie slalomu, który nakłada ograniczenia na ten tor w postaci bramek. Każda bramka ściśle narzuca, z której strony należy ją przejechać, a ominięcie chociaż jednej z nich powoduje dyskwalifikację zawodnika.

Zdefiniowany przez nas problem jest interdyscyplinarny - z pogranicza fizyki i informatyki. Do dobrego zrozumienia zjawisk zachodzących na stoku narciarskim cenne jest też posiadanie własnych doświadczeń z jazdy po trasach slalomu. Wymagania te powodują, że problem nie jest trywialny do rozwiązania i w celu badania go nieodłączne są osoby o różnych kompetencjach.

Obecnie nie udało nam się znaleźć publicznie dostępnych prac, które podchodziłyby do rozwiązania tego praktycznego problemu. Zdajemy sobie sprawę, że problem jest bardzo złożony i próby jego rozwiązania to tak naprawdę rozwiązanie uproszczone tego problemu. Dodatkowo, uwzględnić trzeba fakt, że wiele zmiennych występujących w równaniach wpływa na siebie nawzajem, powodując zmiany niekoniecznie widoczne natychmiast. Może to na przykład sprawiać, że niewielka zmiana dokonana na początku jazdy może mieć znaczący wpływ na ostateczny wynik, co znacznie utrudnia wszelką predykcję na temat wpływu zmian. Aby rozwiązać problem, stworzyłyśmy fizyczny model narciarza - zamodelowany jako punkt materialny o konfigurowalnych parametrach, co umożliwia porównanie wyników np. dla zawodników o różnych masach. Potraktowanie narciarza jako punktu materialnego jest pierwszym z zastosowanych uproszczeń, które zdecydowałyśmy się przyjąć w naszym rozwiązaniu. Stok modelowany jest jako płaszczyzna o zadanym kącie nachylenia, na której za pomocą współrzędnych oznaczamy miejsce występowania bramek. Dużym wyzwaniem było dobranie przybliżenia trasy przejazdu, aby umożliwić wystarczająco łatwe obliczenia i jednocześnie nie tracąc zbytnio na dokładności oddania realnej trasy. Łamana, którą wybrałyśmy jako rozwiązanie spełniające obydwa te wymagania, jest wystarczająco dobrym przybliżeniem jeśli narzucimy na nią dodatkowe ograniczenia jak eliminacja ostrych kątów załamania.

Kluczową częścią naszego rozwiązania jest wykorzystanie algorytmu genetycznego do wybrania pewnego lokalnego optimum trasy, a następnie przeprowadzamy lokalną optymalizację celem wygładzenia znalezionego rozwiązania. Aby przyspieszyć obliczenia, zastosowałyśmy architekturę opartą o rozproszonych klientów wykonujących obliczenia i raportujących do głównego serwera. Na podstawie zebranych danych od klientów, serwer jest w stanie dostarczyć rozwiązanie szybciej oraz można mieć większą pewność, iż jest ono jeśli nie optymalne, to bardzo bliskie optymalnego. Obliczenia wykonywane są w środowisku przeglądarek internetowych w języku JavaScript.

Otrzymane rozwiązanie może mieć zastosowanie nie tylko w celu znajdowania optymalnej trasy przejazdu po zadanym slalomie. Przykładem może być wsparcie dla trenerów ustawiających takie slalomy w postaci aplikacji podpowiadającej gdzie ustawić kolejną bramkę, aby nie było problemów z jej przejechaniem. Dodatkowo, dokładając moduł wyliczający naprężenia i siły działające na stawy kolanowe, można by zredukować negatywny wpływ niefortunnie ustawionych bramek, powodujących wyjątkowe przeciążenia w kolanach, wykrywając to i przestawiając bramki.

1.1. Cele pracy

Celem poniższej pracy jest zapoznanie studentów z systemem LaTeX w zakresie umożliwiającym im samodzielne, profesjonalne złożenie pracy dyplomowej w systemie LaTeX.

1.2. Zawartość pracy

W rodziale **??** przedstawiono podstawowe informacje dotyczące struktury dokumentów w LaTeXu. Alvis [3] jest językiem

2. Wprowadzenie teoretyczne

W rozdziale tym przedstawiono informacje .

2.1. Fizyczny model narciarza

2.2. Metody numeryczne rozwiązywania równań różniczkowych

2.3. Optymalizacja

W tym podrozdziale opisane są metody optymalizacji użyte w zaproponowanym rozwiązaniu, czyli algorytm ewolucyjny oraz algorytm optymalizacji lokalnej - Hill climbing.

Zadaniem optymalizacji jest przeszukanie przestrzeni rozwiązań w celu znalezienia takiego, które jest najlepsze. Zatem mając daną funkcję, nazywaną funkcją celu, która każdemu punktowi reprezentującemu rozwiązanie problemu, poszukujemy takiego, dla którego wartość tej funkcji będzie jak najmniejsza (bądź jak największa). Trudność w znalezieniu takiego rozwiązania zależy od charakteru funkcji celu, a czasem także od nieznajomości jej analitycznej postaci.

Optymalizacja lokalna i globalna

W przypadku funkcji z jednym optimum do znalezienia najlepszego rozwiązania wystarczy przeszukiwanie lokalne. Polega ono na iteracyjnym sprawdzaniu rozwiązań w najbliższej przestrzeni i wprowadzaniu lokalnych zmian, aby w końcu znaleźć rozwiązanie najlepsze w okolicy tzw. optimum lokalne. Jeśli wiemy, że istnieje tylko jedno takie optimum, możemy mieć pewność, że znalezione rozwiązanie jest najlepszym w całej przestrzeni rozwiązań. Przykładami optymalizacji lokalnych są:

– hill climbing

– przeszukiwanie tabu

Jeśli natomiast funkcja celu posiada wiele optimów lokalnych (tzw. funkcja wielomodalna) to optymalizację nazywamy optymalizacją globalną. Jeśli zadanie jest ciągłe, a więc niemożliwe jest przeszukanie całej przestrzeni rozwiązań, nigdy nie możemy być pewni, że zastosowany algorytm optymalizacji da nam rozwiązanie najlepsze - być może będzie to tylko minimum lokalne a nie globalne. Nie mając takiej pewności nie wiemy kiedy należy zatrzymać algorytm. Z tego powodu stosuje się parametr sterujący czasem trwania obliczeń, kosztem mniejszej pewności co do poprawności rozwiązania możemy otrzymać krótszy czas optymalizacji i odwrotnie.

2.3.1. Algorytm ewolucyjny

Algorytm ewolucyjny jest przykładem algorytmu optymalizacyjnego, przeszukującego przestrzeń rozwiązań w celu znalezienia najlepszego rozwiązania problemu. Algorytm ten oparty jest na obserwacjach środowiska i przystosowywania się organizmów do jego warunków. Wiele terminów zapożyczonych jest zatem z genetyki.

Podstawą całego algorytmu jest populacja osobników, z których każdy reprezentuje rozwiązanie problemu. Populacja ta zmienia się wraz z działaniem algorytmu. Ewolucja zakłada, że populacja będzie się składać z coraz lepiej przystosowanych osobników. Przystosowanie to jest obliczane za pomocą wcześniej określonej funkcji oceniającej jakość danego osobnika, czyli jak dobre jest rozwiązanie reprezentowane przez niego. Przystosowanie jest wartością liczbową obliczoną za pomocą tej funkcji przystosowania.
Funkcja przystosowania określa wartość przystosowania osobnika na podstawie jego fenotypu, który jest tworzony z genotypu. Genotyp określa zestaw cech danego osobnika i składa się z chromosomów (najczęściej z jednego). Natomiast każdy z chromosomów składa się z elementarnych jednostek - genów.

Schemat działa algorytmu ewolucyjnego

Algorytm ewolucyjny rozpoczyna się poprzez wygenerowanie populacji bazowej oraz obliczenie przystosowania jej osobników. Przeważnie osobniki te generowane są całkowicie losowo, ale można także wprowadzić konkretne osobniki np. o znanym dobrym przystosowaniu do środowiska.

Główna część algorytmu opiera się na powtarzaniu pętli, w której wykonywane są kolejno:

– reprodukcja

– operacje genetyczne

– ocena

– sukcesja

Często reprodukcję i sukcesję łączy się pod nazwą selekcja.

Reprodukcja powoduje powielenie losowo wybranych osobników z populacji. Prawdopodobieństwo wybrania osobnika do powielenia najczęściej jest proporcjonalne do jego przystosowania. Może się zdarzyć, że dany osobnik zostanie wybrany więcej niż raz, a także, że nie zostanie wybrany ani razu.
Następnie na tych kopiach przeprowadzane są operacje genetyczne powodujące zmiany w genotypie osobników. Wyróżniamy dwie podstawowe operacje:

– mutacja

– krzyżowanie

Zadaniem mutacji jest losowe zmodyfikowanie genów w genotypie.
Krzyżowanie, zwane także rekombinacją (ang. *crossover*), działa na co najmniej dwóch osobnikach i na podstawie ich genotypu tworzy jeden lub więcej osobników potomnych. Chromosomy rodzicielskie są mieszane w celu otrzymania nowych genotypów dla osobników potomnych.

W wyniku operacji genetycznych powstają nowe osobniki, które wchodzą w skład populacji potomnej. Każdy z tych osobników jest oceniany za pomocą funkcji przystosowania. Porównując jakość osobników z populacji bazowej oraz potomnej dokonuje się sukcesji, czyli wyboru osobników z tych populacji (czasem wyłącznie z populacji potomnej) i tworzy nową populację bazową.

Zakończenie działania algorytmu przeważnie opiera się na badaniu funkcji przystosowania całej populacji. Jeśli wartość przystosowania populacji nie jest zróżnicowana mówimy o stagnacji algorytmu i może być to wskazaniem

do zakończenia działania algorytmu. Czasem jednak oczekuje się aż przystosowanie to będzie wystarczająco duże, żeby stwierdzić, że znalezione rozwiązanie jest bardzo dobre. Przeważnie jednak nie znamy nawet przybliżonej wartości jakości rozwiązania, więc nie możemy stwierdzić kiedy przystosowanie jest odpowiednie i czy nie może się jeszcze znacznie poprawić.

Kodowanie osobników

W przypadku algorytmów genetycznych, będących szczególnym przypadkiem algorytmów ewolucyjnych, do kodowania osobników stosuje się kodowanie binarne chromosomów. Pojedynczy bit reprezentuje zatem gen należący do chromosomu.
W takim przypadku mutacja wykonywana jest na każdym genie osobno z pewnym prawdopodobieństwem, jeśli do niej dochodzi, zmienia się wartość bitu na przeciwną. W krzyżowaniu wybiera się dwa osobniki rodzicielskie, których chromosomy rozcinane są na dwie części i łączone ńa krzyż". Miejsce przecięcia jest losowane z rozkładem równomiernym.

W algorytmach ewolucyjnych porzuca się kodowanie binarne - chromosom składa się z jednej lub więcej liczb stanowiących cechy osobnika.
Mutacja takiego osobnika najczęściej odbywa się poprzez losową zmianę każdej z wartości genów chromosomu. Do krzyżowania wybiera się dwa osobniki, z których dla każdej pary odpowiadających genów wyciągana jest średnia i tak otrzymane wartości genów tworzą genotyp nowego osobnika.

Typy algorytmów ewolucyjnych

Algorytmy ewolucyjne wywodzą się z kilku osobnych nurtów zajmujących się tą tematyką, więc istnieje wiele podobnych schematów. Najlepiej traktować algorytmy ewolucyjne jako metaheurystykę - określony jest pewien szkic algorytmu, który można dostosowywać do konkretnego rozwiązania. W tym podrozdziale opisane są podstawowe i najbardziej popularne schematy postępowania oparte o algorytmy ewolucyjne.

Prosty algorytm genetyczny Prosty algorytm genetyczny został zaproponowany w roku 1975 przez John'a Holland'a.

Mając populację bazową P^t dokonujemy reprodukcji tej populacji, tworząc populację tymczasową T^t składającą się z takiej samej liczby osobników. Wybierani są oni z prawdopodobieństwem proporcjonalnym do ich przystosowania z populacji bazowej. Na populacji tymczasowej dokonujemy operacji genetycznych (mutacji i krzyżowania). Do krzyżowania wybierane są rozłączne pary osobników i z pewnym prawdopodobieństwem p_c zachodzi ich skrzyżowanie. Jeśli doszło do powstania osobników potomnych zastępują one osobniki rodzicielskie. Następnie na tak otrzymanej populacji tymczasowej dochodzi do mutacji osobników i otrzymania populacji potomnej O^t. Ta populacja staje się w następnej iteracji algorytmu nową populacją bazową.
Zatrzymanie algorytmu może być dokonane jeśli np.:

- wykonano określoną z góry liczbę iteracji
- znaleziono osobnika o wystarczająco wysokiej wartości przystosowania

W tej wersji algorytmu często pętlę algorytmu nazywa się generacją, a każdą populację P^t w chwili t pokoleniem.

Strategia (1+1) Strategia (1+1) jest podstawową strategii ewolucyjnych. W algorytmie tym mamy do czynienia z populacją składającą się z tylko jednego osobnika posiadającego jeden chromosom. W każdej pętli algorytmu dokonuje się mutacji tego chromosomu, co powoduje powstanie nowego osobnika. Osobnik ten jest poddawany

ocenie, a następnie dokonuje się wyboru lepszego z dwóch istniejących osobników i tego pozostawia w populacji. W mutacji dodaje się do każdego genu chromosomu losową modyfikację rozkładem normalnym:

$$Y_i^t = X_i^t + \sigma\xi_{N(0,1),i} \tag{2.1}$$

Wartość σ będzie powodowała większe lub mniejsze zmiany w chromosomie. Jeśli chcemy przeszukać przestrzeń, powinniśmy zwiększać jej wartość, co jest pożądane zwłaszcza w początkowej fazie działania algorytmu. Natomiast, aby znaleźć jak najlepsze rozwiązanie, wiedząc że obecne rozwiązanie jest już bardzo bliskie najlepszemu, możemy zmniejszać wartość σ przeszukując tylko najbliższą przestrzeń.
Do wyznaczania σ powstał następujący algorytm zwany regułą 1/5 sukcesów:

1. Jeśli przez kolejnych k pętli algorytmu mutacja powoduje powstanie lepszego osobnika w więcej niż 1/5 wszystkich mutacji, to zwiększamy σ: $\sigma' = c_i\sigma$. Wartość c_i wyznaczona empirycznie wynosi $\frac{1}{0.82}$
2. Gdy dokładnie 1/5 kończy się sukcesem, wartość σ pozostaje bez zmian.
3. Jeśli nie zachodzi żadne z powyższych wartość σ jest zmniejszana: $\sigma' = c_d\sigma$. Gdzie c_d powinna wynosić 0.82

Strategia (μ + λ) Strategia (μ + λ) jest rozwinięciem strategii (1+1). μ oznacza ilość osobników w populacji początkowej, a λ ile osobników jest reprodukowanych i poddawanych operacjom genetycznym. Dodatkowo, zamiast reguły 1/5 sukcesów wprowadzono mechanizm samoczynnej adaptacji zasięgu mutacji, a także wprowadzono operator krzyżowania.

Oznaczenie μ + λ oznacza, że po wygenerowaniu populacji potomnej wybierane jest μ najlepszych osobników do nowej populacji bazowej - zarówno spośród populacji potomnej, jak i starej populacji bazowej zawierającej łącznie μ + λ osobników.

W strategii tej ważne jest też kodowanie, do którego dodatkowo dołożono również chromosom przechowujący wektor σ zawierający wartości odchyleń standardowych, które wykorzystuje się w trakcie mutacji.
Po wylosowaniu wartości zmiennej losowej o rozkładzie normalnym ($\xi_{N(0,1)}$) dla każdego elementu wektora σ losujemy jeszcze jedną zmienną losową o rozkładzie normalnym ($\xi_{N(0,1),i}$) i oblicza nowe wartości odchyleń z wektora σ:

$$\sigma_i' = \sigma_i e^{(\tau'\xi_{N(0,1)} + \tau\xi_{N(0,1),i})} \tag{2.2}$$

Gdzie τ oraz τ' są parametrami algorytmu, a ich wartości powinny wynosić:

$$\tau = \frac{K}{\sqrt{2n}} \tag{2.3}$$

$$\tau' = \frac{K}{\sqrt{2\sqrt{n}}} \tag{2.4}$$

Mając dane nowe wartości odchyleń standardowych możemy obliczyć nowe wartości genów korzystając ze wzoru:

$$X_i' = X_i + \sigma_i'\xi_{N(0,1),i} \tag{2.5}$$

gdzie $\xi_{N(0,1),i}$ jest nową losową wartością.

Algorytm ewolucyjny wybiera osobniki lepiej przystosowane, a więc te, które posiadają także lepsze wartości odchyleń standardowych. Powoduje to naturalną selekcję, doprowadzającą do samoczynnej adaptacji odchyleń standardowych stosowanych w trakcie mutacji.

Krzyżowanie występuje w tym algorytmie pod nazwą rekombinacja. Najczęściej sprowadza się do uśrednienia lub wymianie wartości wektorów, także wektora σ.

Strategia (μ, λ) Strategia ($\mu + \lambda$) posiada pewne wady, które postanowiono spróbować wyeliminować za pomocą nowej strategii (μ, λ). Poprzedni algorytm sprawia problemy jeśli w populacji pojawia się osobnik o wysokiej wartości przystosowania, ale posiadający zbyt duże (albo zbyt małe) wartości odchyleń standardowych. Usunięcie takiego osobnika z populacji często nie jest procesem krótkotrwałym, gdyż wpływa on na powstające potomstwo, przekazując mu podobne do jego, nieodpowiednie wartości odchyleń.
W nowej strategii wprowadzono zmianę, która powoduje, że osobniki rodzicielskie nie są nigdy brane do kolejnej populacji bazowej. Podczas selekcji korzysta się zatem tylko z powstałej populacji potomnej, z niej wybierając osobniki do populacji bazowej w kolejnej iteracji.

2.3.2. Hill climbing

Algorytm hill climbing jest jedną z metod przeszukiwania lokalnego. W każdej iteracji zmieniając wartość jednej ze zmiennych rozwiązania sprawdzana jest wartość funkcji celu dla nowego rozwiązania i jeśli wartość ta jest lepsza od dotychczas najlepszej znalezionej, zapamiętujemy zmienione rozwiązanie. Dopóki zmiany powodują poprawę rozwiązania, algorytm nie jest zatrzymywany. Na końcu wiemy, że znalezione rozwiązanie jest rozwiązaniem lokalnie optymalnym.
Przeszukiwanie przestrzeni dyskretnej sprowadza się do sprawdzenia rozwiązań najbliższych obecnemu i wybieranie tego rozwiązania, którego wartość obliczona za pomocą funkcji celu jest najlepsza. Jeśli wśród sąsiadów nie ma już lepszego rozwiązania, możemy zakończyć przeszukiwanie.
W przestrzeni ciągłej konieczne jest dobranie kroku, który wyznacza punkty przeszukiwane w okolicy w trakcie każdej iteracji. Dodatkowo wykorzystywane jest tzw. przyspieszenie (ang. *acceleration*), które wyznacza pięciu możliwych kandydatów na lepsze rozwiązania. Najczęściej przyspieszenie to wynosi 1.2, a wartość kroku jest osobna dla każdej zmiennej rozwiązania i często wynosi na początku 1. Zatem za każdym razem obliczane są następujące współczynniki: -acceleration, -1/acceleration, 0, 1/acceleration, acceleration. Następnie współczynniki mnożone są przez krok (step) i dodawane do obecnie analizowanej zmiennej i wybierane jest najlepsze z pięciu rozwiązań. Wartość kroku jest indywidualna dla każdej zmiennej. Po wybraniu najlepszego rozwiązania uaktualniana jest wartość tego kroku - krok mnożony jest przez odpowiedni współczynnik, ten który był dobrany wcześniej do znalezienia tego najlepszego rozwiązania. Algorytm zatrzymywany jest jeśli zmiana żadnej ze zmiennych nie przynosi już poprawy rozwiązania, czasem również jeśli ta zmiana jest już bardzo mała - wprowadzany jest parametr ϵ wyznaczający tę różnicę.

2.4. Uczenie maszynowe

Uczeniem się systemu jest każda autonomiczna zmiana w systemie zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania. (Cichosz)

Program się uczy z doświadczenia E dla zadań T i miary jakości P jeśli jego efektywność w zadaniach z T mierzona P wzrasta z doświadczeniem E. (Mitchell)

Istnieje wiele rodzajów uczenia maszynowego. Podstawowy podział wynika z rodzaju informacji trenującej na:

– uczenie z nadzorem

– uczenie bez nadzoru

W uczeniu się z nadzorem źródłem informacji trenującej jest nauczyciel. Od niego otrzymuje uczeń informację jakie zachowanie jest pożądane. Natomiast w przypadku uczenia bez nadzoru uczeń dowiaduje się o skuteczności swojego działania obserwując wyniki - nazywa się to czasem wbudowanym nauczycielem.

Istnieją jeszcze dwie grupy, które trudno zakwalifikować do powyższych:

– uczenie się na podstawie zapytań

– uczenie się przez eksperymentowanie

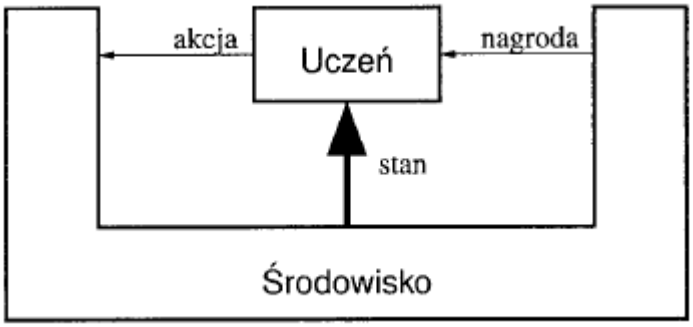
– uczenie się ze wzmocnieniem

Do pierwszej z nich należą algorytmy polegające na zadawaniu pytań przez ucznia nauczycielowi. Natomiast do drugiej te, w których uczeń gromadzi swoje doświadczenia obserwując konsekwencje swojego działania w środowisku. Uczenie się ze wzmocnieniem jest podobne do tej metody, ale dodatkowo istnieje krytyk, który służy jako dodatkowe źródło informacji trenującej. Jego zadaniem jest karanie bądź nagradzanie ucznia za jego zachowanie. Uczeń nie dowiaduje się co ma robić, ale jak wartościowe jest dane działanie.

Czasem granice pomiędzy tymi grupami są nieostre i przynależność algorytmu do jakiejś grupy może zależeć wyłącznie od punktu widzenia.

2.4.1. Uczenie się ze wzmocnieniem

W przypadku uczenia się ze wzmocnieniem zadaniem ucznia jest obserwacja stanów środowiska, wykonywanie akcji oraz obserwowanie efektów tych akcji poprzez wartość otrzymywanego wzmocnienia jako rzeczywistoliczbowej nagrody. Tak jak zostało to napisane wcześniej, w tym przypadku nie mówimy o nauczycielu, ale o krytyku, który wartościuje zachowanie poprzez dostarczanie wzmocnienia. Zadaniem ucznie jest odnalezienie takiego zachowania, które przyniesie mu jak największą nagrodę. Najczęściej uczeń nie ma pojęcia o tym jakie jest środowisko, często niedeterministyczne, dlatego musi wchodzić w interakcję z nim, aby je poznać.



W każdym kroku uczeń jest w określonym stanie środowiska. Decydując się na określoną akcję otrzymuje informację o nowym stanie, w którym znajduje się po wykonaniu tej akcji oraz o nagrodzie (wzmocnieniu) jaką otrzymuje za swoje działanie. Uczeń obserwując nagrody otrzymywane za swoje zachowanie może uczyć się jak postępować, aby były one jak najwyższe.

Schemat algorytmu przedstawia się następująco:
Dla kolejnych kroków czasowych t:

1. obserwujemy stan x_t
2. wybieramy akcję a_t możliwą do wykonania w stanie x_t
3. wykonujemy akcję a_t
4. obserwujemy wzmocnienie r_t i następny stan x_{t+1}
5. uczymy się na podstawie doświadczenia (x_t, a_t, r_t, x_{t+1})

Wybór akcji w kroku 2. dokonywany jest autonomicznie przez ucznia. Natomiast stan, do którego przechodzi po wykonaniu akcji jest określony przez środowisko na podstawie stanu poprzedniego oraz wykonanej akcji. Warto jednak zwrócić uwagę na fakt, że środowisko może być stochastyczne - wykonanie dwa razy tej samej akcji może dawać różne rezultaty. Poza tym, przeważnie środowisko jest nieznane uczniowi, stąd konieczność podejmowania prób i błędów poprzez wykonywanie różnych akcji. Jednocześnie, uczeń nie może wpływać na środowisko w żaden sposób.

Strategia maksymalizacji nagród

Nauka ucznia oparta jest na nagrodach, które otrzymuje za swoje działania. Musi znaleźć on najlepszą strategię wyboru akcji, aby uzyskiwać jak najlepsze nagrody. Najczęściej uczeń próbuje maksymalizować swoje nagrody długoterminowo. Strategia ta polega na tym, że nagrody za poprawne działanie mogą przyjść wiele kroków później niż wtedy gdy ono zostało wykonane. Strategia ta nazywana jest uczeniem się z opóźnionym wzmocnieniem. W uczeniu z natychmiastowym wzmocnieniem interesuje nas tylko maksymalizacja nagród tuż po danym zachowaniu. Nie jesteśmy wtedy w stanie brać pod uwagę tego, jakie w przyszłości mogą być jego skutki.
W przypadku opóźnionego wzmocnienia wprowadza się współczynnik dyskontowania $\gamma \in [0, 1]$. Zadaniem ucznia jest zmaksymalizowanie zdyskontowanej sumy nagród:

$$E[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{2.6}$$

Im współczynnik γ jest bliższy 0, tym bardziej maksymalizuje się tylko natychmiastowe nagrody. Jeśli $\gamma = 1$ to maksymalizowana jest suma wszystkich otrzymanych nagród.

Zadania epizodyczne

W niektórych przypadkach działania ucznia można łatwo wydzielić na niezależne epizody, z których każdy trwa najczęściej skończoną liczbę kroków. Rozdział ten jest kierowany tym, że każda z tych prób jest oceniania osobno. Zatem maksymalizujemy kryterium jakości w każdej próbie oddzielni. Zatem w równaniu 2.6 musimy zastąpić sumę nieskończoną otrzymując:

$$E[\sum_{t=0}^{n-1} \gamma^t r_t] \tag{2.7}$$

Wartość n to liczba kroków epizodu. Powyższe równanie można traktować tak naprawdę jako szczególny przypadek równania 2.6. Zakładając, że w ostatnim kroku wchodzimy do stanu, w którym jedyna możliwa akcja prowadzi z powrotem do niego, a nagroda w tym stanie wynosi 0, otrzymujemy dokładnie powyższe równanie. W równaniu tym zmienna r_t powinna być dla uściślenia zastąpiona przez $r_{i,t}$, ponieważ wartości wzmocnienia mogą być różne w każdej próbie i.

Istnieją także dwa szczególnego rodzaju typy zadań epizodycznych, które nazywane są zadaniami do-sukcesu lub do-porażki. Zakończenie każdej próby kończy się odpowiednio w każdym typie sukcesem lub porażką. W przypadku zadań do-sukcesu chcemy, aby w każdym epizodzie w jak najmniejszej liczbie kroków osiągnąć pewien pożądany stan, co powoduje osiągnięcie sukcesu i zakończenie tej próby. Odpowiednio dla zadań do-porażki jak najbardziej chcemy odwlec moment przejście do niepożądanego stanu, który oznacza porażkę.

Algorytm uczenia się strategii

Q-learning. Algorytm Q-learning jest najczęściej stosowanym algorytmem do uczenia się optymalnej strategii. Uczenie się polega na oszacowaniu optymalnej funkcji wartości akcji. W każdy kroku czasowym obliczana jest wartość następującego wyrażenia, które nazywane jest błędem:

$$\Delta = r_t + \gamma \max_a Q_t(x_{t+1}, a) - Q_t(x_t, a_t); \tag{2.8}$$

r_t to wartość wzmocnienia w tym kroku czasowym, następny człon określa maksymalną wartość funkcji Q dla stanu, w którym znajdzie się uczeń po wykonaniu wybranej wcześniej przez siebie akcji. Tę wartość mnożymy przez współczynnik dyskontowania γ i od tak obliczonej liczby odejmujemy obecną wartość funkcji dla obecnego stanu i wybranej akcji.
Aktualizacja wartości funkcji odbywa się w następujący sposób:

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha\Delta \tag{2.9}$$

Dwa warunki dotyczące wartości α powinny być spełnione:

$$\sum_{i=1}^{\infty} \frac{1}{\alpha_i(x, a)} = \infty \tag{2.10}$$

$$\sum_{i=1}^{\infty} \frac{1}{\alpha_i^2(x, a)} < \infty \tag{2.11}$$

Jednak najczęściej w praktyce rzadko stosuje się do tych warunków.

Warto zwrócić uwagę na fakt, że nie określono tu w jaki sposób uczeń rzeczywiście wybiera akcję. Nie musi to być wcale strategia, w której zawsze wybierana jest najlepsza z akcji.

Algorytm Sarsa. Algorytm Sarsa niewiele różni się od algorytmu Q-learning. Jedyną różnicą jest modyfikacja wyrażenia na błąd:

$$\Delta = r_t + \gamma Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t); \tag{2.12}$$

Zamiast wykorzystywać maksymalną wartość funkcji w przyszłym stanie, korzystamy z wartości, która faktycznie zostanie wybrana. Ponieważ wybór ten dokonywany jest dopiero w następnym kroku, ogólny schemat musi być lekko zmodyfikowany. Różnica ta powoduje, że algorytm Sarsa nie posiada własności algorytmu Q-learning dotyczącego wyboru akcji. Maksymalizacja dokonywana jest z użyciem tej samej strategii, z której korzysta algorytm.

Wybór akcji

Tak jak w przypadku algorytmów ewolucyjnych konieczna jest eksploracja środowiska, w którym znajduje się uczeń. Z drugiej jednak strony chcemy jak najszybciej znaleźć optymalne zachowanie, czyli dokonać eksploatacji wiedzy, którą już posiadamy. Pojawia się pytanie w jaki sposób wybierać akcję, aby dobrze poznać środowisko, a jednocześnie nie zmarnować zbyt dużo czasu na bezcelowe przeszukiwanie opcji, które nie dają korzystnego rozwiązania.
W przypadku algorytmów takich jak Q-learning, w których można użyć innej strategii wyboru akcji niż ta, która jest używana w trakcie maksymalizacji funkcji Q, zapewnienie eksploracji można zapewnić stosując jednostajnie losowy wybór akcji.

Najprostszym sposobem wyboru akcji jest wybranie najlepszej z możliwych akcji, jednak metoda ta uniemożliwia eksplorację. Aby to umożliwić, wprowadza się modyfikację, która polega na tym, że co jakiś czas z prawdopodobieństwem ϵ stosuje się wspomniany wcześniej jednostajnie losowy wybór akcji. Strategia ta nazywana jest strategią ϵ-zachłanną. Metoda ta jest znana jako dobrze równoważąca zadania eksploracji i eksploatacji.

Minusem strategii ϵ-zachłannej jest to, że w trakcie eksploracji każda z akcji jest losowana z takim samym prawdopodobieństwem. Aby wyeliminować ten element, stosuje się metody selekcji nazywane *softmax*. Wszystkie akcje mogą zostać wylosowane, ale z prawdopodobieństwem odpowiadającym im dotychczasowej wartości.

Najczęściej korzysta się z rozkładu Gibbs'a lub Boltzmann'a. Prawdopodobieństwo wybrania akcji a w próbie t wynosi:

$$\frac{\epsilon^{Q_t(a)/\tau}}{\sum_{b=1}^{n} \epsilon^{Q_t(b)/\tau'}} \tag{2.13}$$

Parameter τ nazywany jest temperaturą. Im jest ona wyższa, tym wybór dowolnej akcji jest bardziej losowy. W przypadku temperatur bliskich 0 wybór praktycznie jednoznacznie padnie na najbardziej korzystną akcję.

3. Istniejące rozwiązania

W rozdziale tym przedstawiono informacje .

3.1. Rozwiązanie 1

4. Proponowane rozwiązanie

W rozdziale tym przedstawiono informacje .

4.1. Model narciarza i środowiska

W zaproponowanym rozwiązaniu trzeba była podjąć pewne decyzje odnośnie reprezentacji środowiska oraz narciarza. Zostało przyjęte, że stok traktowany jest jako płaszczyzna, która jest nachylona pod określonym przez stałą α kątem do powierzchni ziemi. Założenie co do płaskiej powierzchni jest tylko ograniczeniem przyjętym do testów. Umożliwia to łatwiejszą analizę wyników niezaburzonych zmianami nachylenia terenu. Jednak stworzony program może zostać zmodyfikowany tak, aby zamodelować bardziej skomplikowaną powierzchnię. Narciarz traktowany jest jako punkt materialny o masie m.

4.2. Opis matematyczny modelu

m - masa
g - współczynnik grawitacji
α - kąt nachylenia powierzchni stoku do powierzchni ziemi

Siła grawitacji działająca na obiekt o masie m:

$$Q = mg \tag{4.1}$$

Siła ta może zostać rozłożona na dwie siły składowe względem powierzchni stoku. Narciarz poruszający się w dół stoku przypomina klasyczny przykład punktu materialnego staczającego się po równi pochyłej.
Składowa siły grawitacji równoległa do powierzchni stoku:

$$Q_a = mg \sin \alpha \tag{4.2}$$

jest to siła ściągająca narciarza w dół stoku.
Składowa siły grawitacji prostopadła do powierzchni stoku:

$$F_n = mg \cos \alpha \tag{4.3}$$

Wartość tej siły wpływa na wartość siły tarcia działającej na narciarza:

$$F_f = \mu F_n = \mu mg \cos \alpha \tag{4.4}$$

Oprócz siły tarcia uwzględniamy również inną siłę oporu jaką jest siła oporu powietrza. Zależy ona od prędkości poruszania się obiektu. Prędkość będziemy wyrażać jako pierwszą pochodną położenia - $\dot{x}$:

$$F_d = k_1\dot{x} + k_2\dot{x}^2 \tag{4.5}$$

Założenia, które narzucane są na stałe k_1 oraz k_2:

$$\begin{cases} k_2 = 0 & v \leq B \\ k_1 = 0 & v \geq B \end{cases} \tag{4.6}$$

gdzie B:

$$B = 4\frac{m}{s} \tag{4.7}$$

Ograniczenie to zostało zaczerpnięte z pracy ???? (Aerodynamic-Drag) i opisane w rozdziale ????

Współczynnik k_1 ....

$$k_1 = ... \tag{4.8}$$

Współczynnik k_2 został opisany w ????

$$k_2 = \frac{1}{2}C\rho A \tag{4.9}$$

gdzie:

C - współczynnik oporu

ρ - gęstość powietrza

A - powierzchnia przednia narciarza prostopadła do kierunku przepływu, a zatem prostopadła do wektora prędkości narciarza

4.3. Numeryczne rozwiązanie problemu

Rozpatrując wszystkie siły działające na narciarza równoległe do powierzchni stoku, a więc powodujące jego ruch w dół, otrzymujemy następujące równanie:

$$ma = mgsin\alpha - \mu mgcos\alpha - k_1\dot{x} - k_2\dot{x}^2 \tag{4.10}$$

Wyraźmy teraz przyspieszenie jako drugą pochodną położenia:

$$a = \ddot{x} \tag{4.11}$$

Podstawiając do równania:

$$m\ddot{x} = mgsin\alpha - \mu mgcos\alpha - k_1\dot{x} - k_2\dot{x}^2 \tag{4.12}$$

Zatem po podzieleniu przez m:

$$\ddot{x} = gsin\alpha - \mu gcos\alpha - \frac{k_1}{m}\dot{x} - \frac{k_2}{m}\dot{x}^2 \tag{4.13}$$

Aby rozwiązać to równanie numerycznie, wprowadzamy nową zmienną v (odpowiadającą prędkości):

$$v = \dot{x} \tag{4.14}$$

Otrzymujemy następujące równania:

$$\begin{cases} \dot{v} = gsin\alpha - \mu gcos\alpha - \frac{k_1}{m}v - \frac{k_2}{m}v^2 \\ v = \dot{x} \end{cases} \tag{4.15}$$

Pamiętamy przy tym, że:

$$\begin{cases} k_2 = 0 \quad v \leq B \\ k_1 = 0 \quad v \geq B \end{cases} \tag{4.16}$$

Powyższy układ równań wystarczy wykorzystać w dostępnych w wielu językach funkcjach bibliotecznych do rozwiązywania równań różniczkowych zwyczajnych. W naszym przypadku wykorzystałyśmy funkcję dopri (Numerical integration of ODE using Dormand-Prince RK method) z biblioteki Numeric Javascript.

4.3.1. Rozwiązanie w 3D

Powyższy układ równań opisuje poruszanie się punktu materialnego po równi pochyłej. Jednak w przypadku narciarza przemieszczającego się po stoku musimy uwzględnić również możliwość poruszania się w poprzek stoku, a nie tylko w dół.
Załóżmy, że narciarz porusza się tylko po liniach prostych.
Rozpatrzmy teraz jak będzie wyglądał układ sił działających na narciarza:

Na naszego narciarza działają siły: grawitacji - w kierunku pionowym oraz siły oporu - w kierunku linii jazdy narciarza. Jeśli rozpatrzymy teraz układ współrzędnych zorientowany względem kierunku jazdy, musimy najpierw zrzutować siłę grawitacji otrzymując siłę ściągająca narciarza:

$$g \sin \alpha * sinus \tag{4.17}$$

gdzie sinus to:
Zatem nasze równanie ruchu będzie wyglądało następująco:

$$\ddot{x} = g \sin \alpha * sinus - (\mu F_n + \frac{k_1}{m}\dot{x} + \frac{k_2}{m}\dot{x}^2) \tag{4.18}$$

gdzie F_n to siła nacisku narciarza, która pozostaje taka sama jak w przypadku 2D:

$$F_n = gcos\alpha \tag{4.19}$$

Transformując powyższe równanie na układ współrzędnych zorientowany wzdłuż i wszerz stoku otrzymamy następujące równania:

$$\begin{cases} \ddot{x_x} = (g * \sin \alpha * sinus - (\mu * F_n + \frac{k_1}{m}\dot{x} + \frac{k_2}{m}\dot{x}^2)) * sinus \\ \ddot{x_y} = (g * \sin \alpha * sinus - (\mu * F_n + \frac{k_1}{m}\dot{x} + \frac{k_2}{m}\dot{x}^2)) * cosinus \end{cases} \tag{4.20}$$

Po wprowadzeniu jak poprzednio dodatkowych zmiennych, w tym wypadku prędkości v_x i v_y oraz pamiętaniu o zależności między nimi a pochodną przemieszczenia:

$$\begin{cases} v_x = \dot{x_x} \\ v_y = \dot{x_y} \\ \dot{x} = \sqrt{v_x^2 + v_y^2} \end{cases} \tag{4.21}$$

otrzymujemy:

$$\begin{cases} v_x = \dot{x_x} \\ v_y = \dot{x_y} \\ \dot{v_x} = (g * \sin\alpha * sinus - (\mu * N + \frac{k_1}{m}\dot{x} + \frac{k_2}{m}\dot{x}^2)) * sinus \\ \dot{v_y} = (g * \sin\alpha * sinus - (\mu * N + \frac{k_1}{m}\dot{x} + \frac{k_2}{m}\dot{x}^2)) * cosinus \end{cases} \tag{4.22}$$

$$\begin{cases} v_x = \dot{x_x} \\ v_y = \dot{x_y} \\ \dot{v_x} = gsin\alpha - (\dot{x_x}^2 + \dot{x_y}^2)^{\frac{1}{2}}\kappa\dot{x_y}sgn(\dot{\varphi}) - gsin\alpha\frac{\dot{x_y}^2}{\dot{x_x}^2+\dot{x_y}^2} - \frac{(F_f+F_d)}{m}\frac{\dot{x_x}}{(\dot{x_x}^2+\dot{x_y}^2)^{\frac{1}{2}}} \\ \dot{v_y} = (\dot{x_x}^2 + \dot{x_y}^2)^{\frac{1}{2}}\kappa\dot{x_x}sgn(\dot{\varphi}) + gsin\alpha\frac{\dot{x_x}}{\dot{x_x}^2+\dot{x_y}^2} - \frac{(F_f+F_d)}{m}\frac{\dot{x_y}}{(\dot{x_x}^2+\dot{x_y}^2)^{\frac{1}{2}}} \end{cases}$$

4.4. Optymalizacja toru przejazdu

Aby znaleźć rozwiązanie problemu, należy przyjąć jakiś sposób reprezentacji każdego z rozwiązań. Tor przejazdu narciarza w rzeczywistości to ślad, który pozostawiają narty na śniegu w trakcie przemieszczania się po stoku. Jak opisano w rozdziale 4.2 ????, w celu uproszczenia sposobu przemieszczania się narciarza, zostało zdecydowane, że jako przybliżenie można przyjąć poruszanie się po łamanej. Zatem do reprezentacji rozwiązania można przyjąć zbiór punktów, przez które kolejno przejeżdża narciarz, poruszając się między tymi punktami wyłącznie po linii prostej.

Jednak wciąż takie podejście jest niewystarczające, ponieważ w algorytmach optymalizacyjnych potrzebujemy ściślejszego opisu, aby wiedzieć jak skutecznie przeszukiwać przestrzeń rozwiązań. Zatem w zaproponowanym rozwiązaniu narzucamy z góry co ile metrów w pionie stoku ma pojawić się punkt. Można traktować to jako poziome linie, z której każda wyznacza możliwe położenie pojedynczego punktu. Oprócz tego narzucone zostało, że narciarz musi przejechać jak najbliżej każdej wewnętrznej bramki, co indukuje dołożenie punktów w miejscu każdej wewnętrznej bramki. Warunek ten jest spowodowany tym, że w przeciwnym przypadku algorytm miałby do przeszukania dużo więcej rozwiązań. Opierając się na doświadczeniu z rzeczywistych slalomów, wiadomo, że przejeżdżanie tuż przy bramkach jest najkorzystniejsze.

Zatem pozostaje określić w jaki sposób możemy stwierdzić, że dane rozwiązanie jest najlepsze. W przypadku algorytmów optymalizacyjnych zawsze należy określić funkcję celu. W naszym przypadku interesuje nas, aby narciarz w jak najkrótszym czasie dotarł do mety. Mając dane rozwiązanie w postaci punktów wyznaczających łamaną, obliczamy ile czasu zajmie narciarzowi przejechanie po tej trasie. Im mniejsza wartość tym rozwiązanie jest lepsze, gdyż tym mniej czasu potrzebuje narciarz na prawidłowe pokonanie slalomu.

4.4.1. Algorytm ewolucyjny

Opisana powyżej reprezentacja rozwiązania to w zastosowanym algorytmie ewolucyjnym pojedynczy osobnik, a punkty składające się na to rozwiązanie, a ściślej, ich położenie w pozycji poziomej określają genotyp każdego osobnika. Nie wprowadzamy tu typowego dla algorytmów genetycznych kodowania binarnego, pozycja każdego punktu jest zapamiętana jako wartość rzeczywista.

Dodatkowo do genotypu wchodzi także parametr σ, który w strategiach ewolucyjnych używany jest podczas mu-

tacji tak jak opisano w rozdziale 2 w części o strategiach. Każdemu punktowi przypisana jest osobna wartość σ - odchylenie standardowe.

Zastosowany algorytm opiera się na strategii (μ + λ) opisanej w rozdziale 2. Jako początkową populację wybieramy losowe osobniki - poziome wartości punktów są ograniczone jedynie przez wartości poziome położenia dwóch najbliższych bramek. Jest to kierowane koniecznością zadbania o szybsze znalezienie rozwiązania - zbyt duże odległości można z góry odrzucić opierając się na doświadczeniach z rzeczywistej jazdy narciarza po slalomie. Wielkość populacji bazowej μ jest jednym z parametrów programu, ale najczęściej wartość ta wynosi 30. Wartość parametru λ także jest parametrem, jednak w testach przeważnie użyto wielkości 100.
Szkielet algorytmu zgodny jest z zastosowaną strategią, po wylosowaniu z istniejącej populacji populacji tymczasowej o wielkości λ, dokonuje się na jej osobnikach operacji genetycznych, najpierw krzyżowania, a następnie mutacji na osobnikach otrzymanych z krzyżowania. Kolejnym krokiem jest ocenienie nowych osobników i wybranie spośród nich oraz populacji początkowej osobników o najlepszym przystosowaniu i to one stanowią nową populację bazową.

Krzyżowanie

Aby dokonać krzyżowania potrzebne są pary rodziców dla każdego nowego osobnika. Aby utrzymać wielkość populacji tymczasowej, losujemy (ze zwracaniem) λ par spośród populacji tymczasowej. Krzyżowanie rodziców sprowadza się do obliczenia średniej wartości położenia odpowiadających sobie punktów oraz parametrów σ.

Mutacja

Po krzyżowaniu mamy znowu w populacji tymczasowej λ osobników. Mutacja osobników przeprowadzana jest zgodnie ze strategią - wykorzystywane są wartości odchyleń standardowych odpowiadających kolejnym punktom. Jedynie punkty, które są przy bramkach nie podlegają mutacji. Wynika to z wcześniejszego założenia, że wtedy otrzymamy rozwiązanie najlepsze.

Warunek zakończenia

Warunek zakończenia algorytmu zawsze sprawia wiele problemów. Nie jest łatwo zdecydować na jakiej podstawie zatrzymywać jego działanie. Często korzysta się z informacji o rozrzucie przystosowania w populacji - obliczamy go na podstawie różnicy pomiędzy najlepszym i najgorszym osobnikiem. Jeśli rozrzut ten jest niewielki może oznaczać stagnację algorytmu. Niekoniecznie świadczy to o znalezieniu rozwiązania, ale w połączeniu z dodatkowymi mechanizmami może być skuteczną metodą na decyzję o zakończeniu optymalizacji.
W naszym rozwiązaniu bierzemy zatem również pod uwagę taki wskaźnik jak poprawa najlepszego obecnego rozwiązania. Jeśli przez określoną liczbę iteracji, najczęściej kilka lub kilkanaście najlepsze rozwiązanie nie poprawia się w ogóle, a populacja jest bardzo mało zróżnicowana to jest to znak, że znalezione rozwiązanie powinno być wystarczająco bliskie najlepszemu. Oczywiście sterując liczbą iteracji przez które sprawdzamy zmiany oraz wielkością rozrzutu populacji możemy znajdować lepsze lub gorsze rozwiązania kosztem wydłużenia lub skrócenia czasu działania algorytmu.

4.4.2. Hill climbing

Zastosowanie algorytmu genetycznego sprawdziło się w przypadku problemu narciarza, jednak problemem był długi czas wykonywania się programu. Zwłaszcza słaba poprawa wyników występowała w końcowej fazie działania. Widoczne były niepotrzebne próby przeszukiwania zbyt odległych rozwiązań, a jednak wciąż znalezione rozwiązanie nie było tak dobre jak można by tego oczekiwać. Wiedząc, że rozwiązanie jest już dosyć bliskie najlepszemu można z dużym prawdopodobieństwem założyć, że wystarczy znaleźć rozwiązanie lokalnie optymalne, aby było ono satysfakcjonujące. Oczywiście nie mamy pewności, że będzie globalnie optymalne, ale takiej pew-

ności nie możemy mieć nigdy.
Zatem zastosowanie algorytmu lokalnej optymalizacji powinno pomóc w końcowej fazie poszukiwań. Z tego powodu użyty został algorytm Hill climbing opisany w rozdziale 2. W każdym kroku algorytmu sprawdzane jest czy zmiana pojedynczej zmiennej - w tym wypadku poziomej pozycji punktu przejazdu, daje poprawę wyniku. Jeśli zmiany te nie przynoszą rezultatów, są mniejsze niż narzucony parametr ϵ algorytm zatrzymuje swoje działanie. Wartość ϵ wynosi przeważnie 0.00001. W przypadku parametrów typowych dla tego algorytmu postanowiono wybrać wartości dla przyspieszenia standardowa - 1.2, natomiast dla kroku, mniejszą niż zwykle, bo wynoszącą 0.5. Zmiana ta wynika z założenia, że rozwiązanie nie powinno potrzebować większych zmian, aby znaleźć rozwiązanie jak najlepsze.

4.5. Architektura systemu

5. Wyniki

W rozdziale tym przedstawiono informacje .

5.1. Optymalizacja

5.1.1. Algorytm ewolucyjny

5.1.2. Lokalna optymalizacja

5.2. Uczenie maszynowe

5.3. Podsumowanie

5.4. Optymalizacja toru przejazdu

5.5. Architektura systemu

6. Podsumowanie

W rozdziale tym przedstawiono informacje .

6.1. Podrozdział

Bibliografia

[1] A. DILLER, *LaTeX wiersz po wierszu*, Wydawnictwo Helion, Gliwice, 2000.

[2] L. LAMPORT, *LaTeX system przygotowywania dokumentów*, Wydawnictwo Ariel, Krakow, 1992.

[3] M. SZPYRKA, *On Line Alvis Manual*, AGH University of Science and Technology, 2011, `http://fm.ia.agh.edu.pl/alvis:manual`.