**Analiza algorytmów sterowników dźwigu osobowego pod kątem optymalizacji średnich czasów obsługi pasażerów w różnych, rzeczywistych przypadkach.**

Filip Antoniak (279929)

**1. Wstęp:**

Cel: Istotą tego badania jest przedstawienie różnych podejść – algorytmów i usprawnień pracy windy w różnych scenariuszach oraz zestawienie i omówienie ich zalet i wskazanie potencjalnych usprawnień.

Badanie to zakłada całkowitą ignorancję w stosunku do zagadnień takich jak koszty związane ze zużyciem energii by w pełni skupić się na minimalizowaniu czasu, którzy potrzebny jest na wykonanie zleceń. Dodatkowo warto zaznaczyć, że omawiana winda charakteryzować będzie się zmiennymi parametrami i będzie to pojedynczy dźwig wyposażony w przyciski kierunkowe.

Wszystkie badane algorytmy zostały zaprojektowane i napisane przeze mnie (dotyczy to również nazewnictwa), na podstawie obserwacji, a następnie zweryfikowane.

**Spis treści:**

1. [Wstęp](#Wstęp)
2. [Wprowadzenie do omawianego problemu](#probmelu)
3. [Wprowadzenie do środowisk symulacyjnych](#srodowiska)
4. [Algorytm AZE – Algorytm Zgodnych Ekstremów](#czter)
5. [Algorytm ANC – Algorytm Najbliższego Celu](#piec)
6. [Algorytm APKZ – Algorytm Priorytetu Kolejności Zgłoszeń](#szesc)
7. [Środowisko symulacyjne – pełna losowość + duża intensywność](#siedem)
8. [Środowisko symulacyjne – scenariusz bloku mieszkalnego](#osiem)
9. [Wnioski dotyczące wydajności algorytmów AZE i ANC w środowiskach; losowym i bloku mieszkalnego](#dziewiec)
10. [Algorytmy uczenia maszynowego a optymalizacja łącznego czasu podróży](#dziesiec)
11. [Wnioski ogólne](#jedensaciee)

**2.** **Wprowadzenie do omawianego problemu**

Zagadnienie jest stosunkowo trudne, ponieważ sposób w jaki, w rzeczywistości działa winda jest dość intuicyjny i bliski optymalnemu (w znaczeniu proporcji między czasem przejazdu a kosztami). Warto zwrócić uwagę, że optymalizacja tego procesu, oprócz oczywistych zadań takich jak zmiana algorytmu, dotyczy także wprowadzenia nowych parametrów, z których ten algorytm mógłby korzystać. Wśród takich dodatkowych parametrów można wyznaczyć na przykład *ilość osób jadących na dane piętro*. W idealnym świecie taka opcja mogłaby znaleźć się obok wybieranego piętra na panelu wewnątrz windy. Natomiast w rzeczywistości ludzie wykorzystywaliby to dla swojej korzyści - do wymuszania priorytetu, co nie szłoby w parze z optymalizowaniem czasu przejazdu dla innych użytkowników. Tutaj, rozwiązać problem i zbliżyć inżynierów do „idealnego świata” mogłaby sztuczna inteligencja, badając rzetelność wprowadzonego parametru lub też sama go szacować na podstawie wzrostu wagi pasażerów.

Kolejną potencjalną optymalizacją dla wind obsługujących wielopiętrowe budynki z małą częstotliwością przejazdów jest wprowadzenie piętra postojowego. Zasada działania jest prosta. Zamiast pozostawania na piętrze, które było celem podróży ostatniego pasażera, winda jechałaby na *wyważone* piętro, które byłoby miejscem oczekiwania na nowe zlecenie (o ile w między czasie nie pojawiłoby się nowe). Samo *wyważanie* mogłoby odbywać się statycznie, dynamicznie bądź przy użyciu sztucznej inteligencji, która dostosowywałaby się do nawyków jej użytkowników. Na przykład fakt, że Pani Jasia co wtorek o 7:30 wychodzi na zakupy.

Inna jest problematyka miejsc, gdzie natężenie ruchu pojedynczego dźwigu jest duże; przykładem jest budynek *D1 Politechniki Wrocławskiej,* gdzie można spotkać się z tak zwanymi pasażerami widmo, czyli osobami, którym znudziło się czekanie na windę i wybrali schody, czy też jak niektórzy sądzą zjawami studentów pokonanych przez sesje. Jedno jest pewne, samo istnienie pasażerów widmo jest czymś, co należy wyeliminować, by zminimalizować czasy przejazdów, ponieważ odciąży to kolejkę priorytetów windy o piętra na których nikt nie czeka. Rozwiązaniem tego problemu mógłby być prosty czujnik, który weryfikowałby aktualność zgłoszenia na danych piętrach.

Z tego wynika, że nie ma jednoznacznego przepisu na idealny, uniwersalny sterownik do windy a sama jej praca może być zoptymalizowana.

**3.** **Wprowadzenie do środowisk symulacyjnych**

Obraz zawierający zrzut ekranu, tekst, diagram, design

Opis wygenerowany automatycznie Do przeprowadzenia doświadczeń i badań, wykorzystany został język *Python* oraz jego moduły takie jak *seaborn, matplotlib, scipy, numpy*, a do wizualizacji przebiegów moduł *pygame*. Pasażerowie generowani są losowo, zgodnie z założeniami danych środowisk. Wprowadzony podział pozwala lepiej spojrzeć na badane algorytmy przez pryzmat różnych przypadków ich użycia, dając kompleksowy obraz na ich efektywność i stopień optymalizacji. Każdy algorytm dla danego środowiska ma ustandaryzowane zjawiska losowe, poprzez wprowadzenia *random.seed(42)*.

Rysunek 1: Wizualizacja przebiegów symulacji w bibliotece pygame

**4.** **Zamodelowana rzeczywistość – rzeczywisty algorytm pracy windy**

Nie istnieje jeden wzorzec, standaryzującego prace windy. Na potrzeby tego badania, rzeczywistym algorytmem będzie nazwany ***Algorytm Zgodnych Ekstremów***.

* Algorytm ten jedzie w danym kierunku (inicjalizowanym zależnie od pierwszego chronologicznie zgłoszenia)
* I zabiera wszystkich pasażerów po drodze, jeśli ich kierunek jest zgodny.
* Następnie jedzie w danym kierunku, dopóki nie obsłuży wszystkich pasażerów z ustalonej kolejki.
* Po czym zmienia kierunek.

**pietra\_zgloszenia** - zbiór wezwań na i-te piętra

**pietra\_cele** - zbiór celów podróży

Zmiana kierunku zależy od osiągniecią **skrajności** wyznaczonej poprzez **sume zbiorów pietra\_zgloszenia** i **pietra\_cele**, lub minimalne/maksymalne piętro dostępne w środowisku**.**

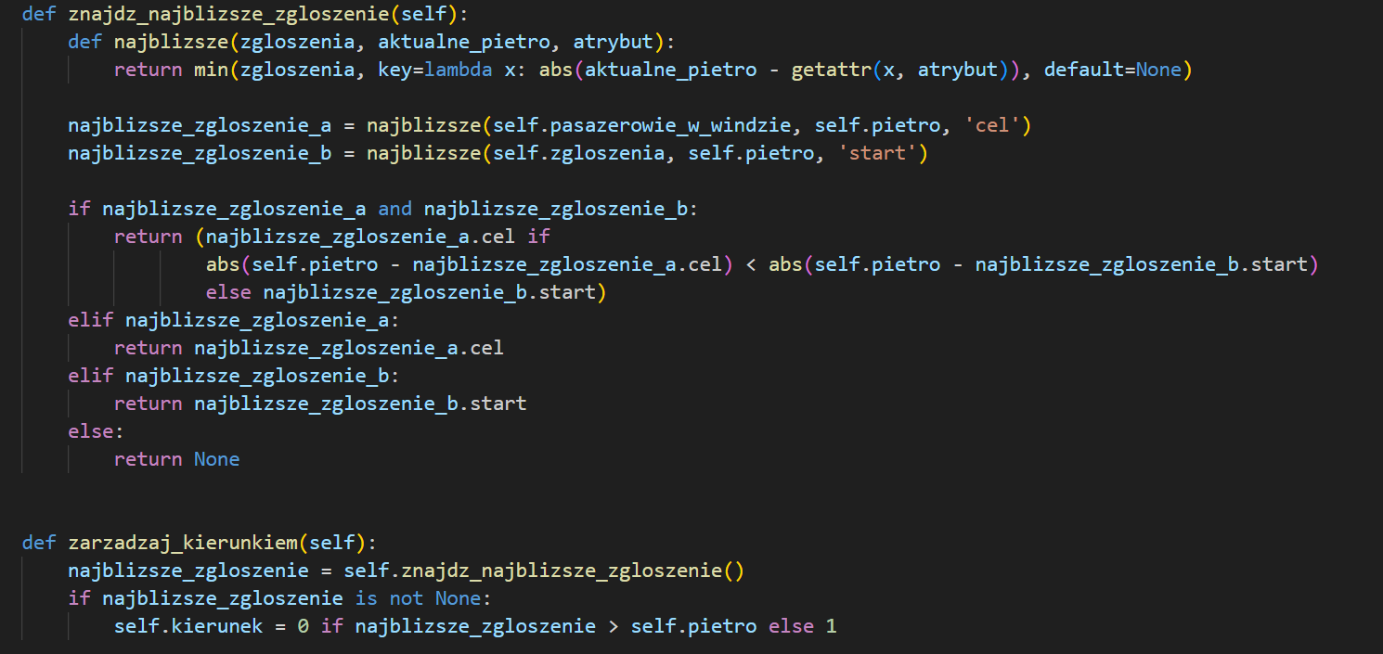
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**5****. Algorytm Najbliższego Celu**

Jest to algorytm, który w inny sposób generuj kolejkę priorytetów pracy windy. Jego działanie polega na znajdywaniu najbliższego, względem pozycji windy celu pasażera wewnątrz windy oraz jeśli to ma sens, najbliższego spośród zgłoszeń. Podobnie jak AZE zabiera wszystkich na swojej drodze, jeśli kierunek się zgadza.

* Algorytm ten jedzie w danym kierunku (inicjalizowanym zależnie od pierwszego chronologicznie zgłoszenia – najbliższego zgłoszenia)
* I zabiera wszystkich pasażerów po drodze, jeśli ich kierunek jest zgodny.
* Następnie kierunek jest ustalany na bazie minimalnej odległości od piętra do: najbliższego celu spośród pasażerów windy oraz o ile winda nie jest pełna najbliższego startu spośród zgłoszeń.



**6.** **Algorytm Priorytetu Kolejności Zgłoszeń**

Jest przykładem złego wykonania sterownika, którego zasadnicza różnica względem AZE polega na ignorowaniu skrajności. Praca windy dyktowana jest przez chronologie zgłoszeń.

* Algorytm ten jedzie w danym kierunku (inicjalizowanym zależnie od pierwszego chronologicznie zgłoszenia)
* I zabiera wszystkich pasażerów po drodze, jeśli ich kierunek jest zgodny.
* Następnie kierunek jest ustalany na bazie najpierw: najwcześniejszego pasażera wewnątrz windy, później, gdy są obsłużeni, zabierani są ci, którzy zgłosili chęć podróży najwcześniej.

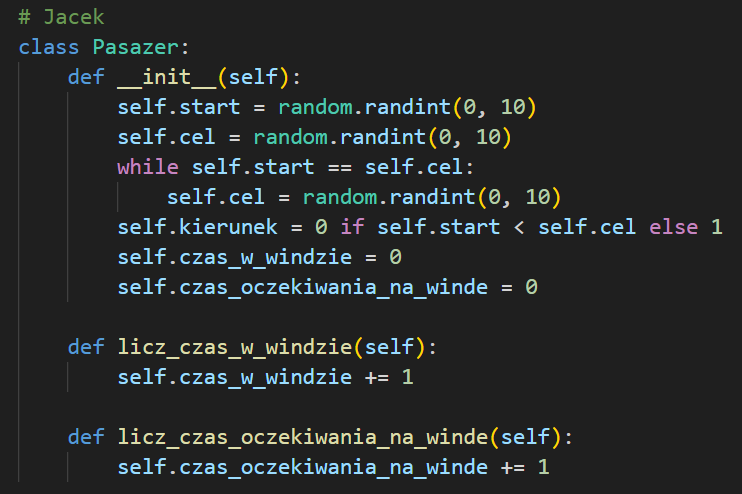
Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**7****. Środowisko symulacyjne – pełna losowość + duża intensywność**

Pierwszym z omawianych przypadków jest środowisko symulacyjne w pełni losowe, to znaczy takie, gdzie oprócz pasażerów początkowych dodawani są losowo nowi pasażerowie co iteracje do czasu obsłużenia 100 tysięcy pasażerów. Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

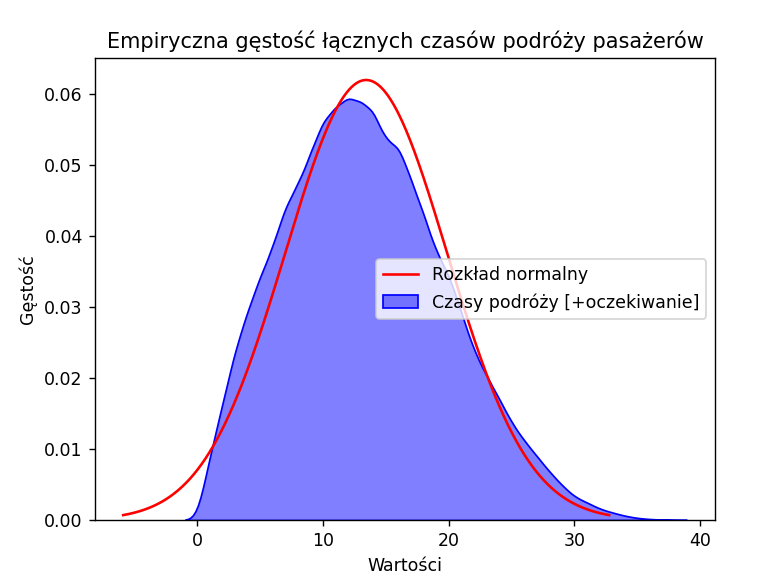
Opis wygenerowany automatycznie Jackowie są inicjalizowani na losowych piętrach z losowymi celami podróży i skrupulatnie liczą czasy swoich podróży.



Losowa natura środowiska pozwala na wykorzystanie narzędzi statystycznych w celu zbadania efektywności i zdiagnozowanie potencjalnie niechcianych odchyleń. Empiryczny rozkładu gęstości sum łącznych czasów podróży, dla próby n = 100.000, powinien dążyć do rozkładu normalnego. Teza ta mówiąca, że *gdy empiryczny rozkład przypomina normalny i brak mu ogona to duża liczba Pasażerów będzie zadowolona*, będzie dodatkowym czynnikiem pomiaru efektywności algorytmów.

Ważnym elementem tego środowiska, jest to, że pojemność wind jest nieskończona. To znaczy nie występuje ograniczenie ilości pasażerów, którą windy mogą zabrać.

**Algorytm Zgodnych Ekstremów w środowisku losowym**



Rysunek 2: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku losowym AZE w zestawieniu z rozkładem normlanym

Rysunek 2. pokazuje empiryczny rozkład gęstości czasów dla 100.000 przebiegów algorytmu AZE w środowisku w pełni losowym. Wskazuje to zgodność z teza, a co za tym idzie względną efektywność algorytmu. Ogon nie jest długi więc ilość przypadków, gdy pasażer musi czekać więcej niż 30 iteracji jest minimalna. Rozkład jest bliski symetrycznemu.

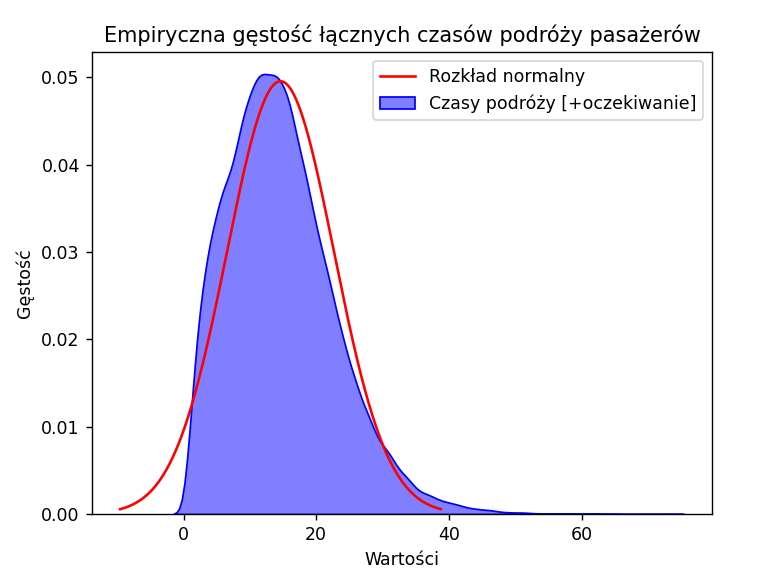
Dodatkowymi parametrami, umożliwiającymi oszacowanie stopnia optymalizacji informacje zbierane przez Jacków.

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100000 |
| Czas [ilość pięter]: | 109941 |
| Średni czas w windzie: | 6.97069 |
| Średni czas oczekiwania na windę: | 6.46927 |
| Średni czas podróży: | 13.43996 |
| Wariancja czasów podróży | 41.4597 |

Tabela 1: Statystyki przebiegów AZE dla 100.000 pasażerów w środowisku w pełni losowym

Dane z Tabeli 1 pokrywają się z odczytami znormalizowanego rozkładu empirycznego, mierząc dodatkowo czasy oczekiwania oraz podróży. Pokazują też dokładną wartość wariancji, która nie wykazuje anomalii np. pasażer czekający 100 iteracji.

**Algorytm Najbliższego Celu w środowisku losowym**



Rysunek 3: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku losowym ANC w zestawieniu z rozkładem normlanym

Rysunek 3. obrazuje efektywność algorytmu ANC. Można zauważyć, że istnieje spora część pasażerów, których czasy podróży są stosunkowo krótkie – lewa część wykresu jest stroma. Wynika to z charakterystyki algorytmu. Nie mniej rozkład rozpina wartości aż do około 70, co jest złe dla docelowej optymalności. Algorytm ten uzyskał rozkład podobny do rozkładu normalnego z wydłużonym ogonem i przechyleniem w lewą stronę.

Jest to algorytm stanowiący potencjalną alternatywę dla AZE.

Tabela 2: Statystyki przebiegów ANC dla 100.000 pasażerów w środowisku w pełni losowym

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100000 |
| Czas [ilość pięter]: | 110379 |
| Średni czas w windzie: | 7.98416 |
| Średni czas oczekiwania na windę: | 6.63116 |
| Średni czas podróży: | 14.61532 |
| Wariancja czasów podróży | 64.8767 |

Analiza danych z Tabeli 2. potwierdza uzyskane wnioski. Algorytm charakteryzuje się dużą wariancją, co nie jest korzystne, gdyż promowane są skrajności takie jak bardzo krótki jak i bardzo długie podróże. Mija się to z zasadą złotego środka. Algorytm ten, choć nie sprawuje się idealnie w środowisku losowym o dużej intensywności, mógłby sprawdzić się w innych warunkach.

**Algorytm Priorytetu Kolejności Zgłoszeń w środowisku losowym**

Obraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznie

Rysunek 4: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku losowym APKZ w zestawieniu z rozkładem normlanym

Rysunek 4. pokazuje, że APKZ uzyskał rozkład znacznie odmienny od normalnego, ogon rozpina się aż do wartości 120 co jest katastrofalnym odchyleniem. Wariancja na poziomie 92 ukazana na Tabeli 3. potwierdza odczyty, mówiące, że algorytm poniósł klęskę w środowisku losowym o dużej intensywności.

Tabela 3: Statystyki przebiegów APKZ dla 100.000 pasażerów w środowisku w pełni losowym

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100000 |
| Czas [ilość pięter]: | 109941 |
| Średni czas w windzie: | 7.01099 |
| Średni czas oczekiwania na windę: | 7.88387 |
| Średni czas podróży: | 14.89486 |
| Wariancja czasów podróży | 91.9107 |

**8****. Środowisko symulacyjne – scenariusz bloku mieszkalnego**

W tym środowisku wprowadzona będzie maksymalna pojemność windy, równa 6 pasażerów. Ruch będzie się odbywał z niską intensywnością, symulowaną na wzór rzeczywistych wzorców ruchu pasażerów w budynku mieszkalnym. To znaczy, znaczna większość podróży odbywać się będzie pomiędzy piętrem *n* a parterem, a ilość pasażerów na zgłoszenie będzie zawierać się pomiędzy 1 a 6, z średnią wynoszącą 2.

P(1) = 0.3, P(2) = 0.25, P(3) = 0.2, P(4) = 0.15, P(5) = 0.05, P(6) = 0.05

Algorytm APKZ będący przykładem niepożądanych zachowań pracy windy, nie będzie omawiany w kontekście tego środowiska, ponieważ już na tym etapie można odrzucić go na podstawie wydajności.

**Algorytm AZE w środowisku bloku mieszkalnego**

Obraz zawierający tekst, diagram, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 5: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku bloku mieszkalnego AZE w zestawieniu z rozkładem normlanym

Tabela 4: Statystyki przebiegów AZE dla 100.000 pasażerów w środowisku bloku mieszkalnego

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100000 |
| Czas [ilość pięter]: | 799921 |
| Średni czas w windzie: | 5.76127 |
| Średni czas oczekiwania na windę: | 4.65831 |
| Średni czas podróży: | 10.41958 |
| Wariancja czasów podróży | 28.8216 |

Jak widać na Rysunku 5. algorytm AZE ponownie pokrywa się z rozkładem normalnym, można zauważyć skokowy charakter, wynikający z natury środowiska – przebiegów od 0 do *n-tego piętra* i od *n-tego piętra* do 0. Niska wariancja jak i brak ogona informują, że algorytm ten dobrze spisuje się w zadanym środowisku.

**Algorytm ANC w środowisku bloku mieszkalnego**

**Obraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 6: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku bloku mieszkalnego ANC w zestawieniu z rozkładem normlanym

Również w tym środowisku, ANC charakteryzują strome brzegi rozkładu jak i długi ogon, sięgający tym razem do wartości 120. Oznacza to, że gdy intensywność ruchu wzrasta, tworzą się problemy związane z nieobsługiwaniem skrajnych przypadków.

Tabela 5: Statystyki przebiegów ANC dla 100.000 pasażerów w środowisku bloku mieszkalnego

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100001 |
| Czas [ilość pięter]: | 801933 |
| Średni czas w windzie: | 5.99015 |
| Średni czas oczekiwania na windę: | 4.6811 |
| Średni czas podróży: | 10.6713 |
| Wariancja czasów podróży | 42.6153 |

**Opymalizacja AZE względem środowiska bloku mieszkalnego**

Kroki podjęte w celu poprawy stopnia optymalizacji:

* Zbadanie i zaimplementowanie punku postoju

Obraz zawierający tekst, linia, diagram, Wykres

Opis wygenerowany automatycznie

Rysunek 7: Zestawienie średnich czasów podróży i wariancji czasów AZE dla różnych punktów postoju.

Jak widać na Rysunku 7. dodanie punktu postoju na piętrze 2 minimalizuje średni czas podróży. Wartość średniego czasu bez punktu postoju wynosi 10.41958 natomiast dla punktu postoju = 2 wynosi 9.87528, daje to około 5.51% poprawę w czasie.

**Optymalizacja ANC w środowisku bloku mieszkalnego**

Dodanie punktów postoju w ANC prezentuje następujące rezultaty

Obraz zawierający tekst, diagram, linia, Wykres

Opis wygenerowany automatycznie

Rysunek 8: Zestawienie średnich czasów podróży i wariancji czasów AZE dla różnych punktów postoju.

Rysunek 8. pokazuje, że ANC osiąga najlepsze rezultaty pod kątem czasu przejazdu, gdy punkt postoju ustawiony jest na wartość 0. Rezultaty te są lepsze o 3.735%. Średni czas wyniósł **10.2273**. Niemniej, jest to uzyskane kosztem znacznego wzrostu wariancji.

Potencjalne kolejne optymalizacje mogą dotyczyć dodania wspomnianych parametrów takich jak ilość osób na zgłoszenie by jeszcze lepiej ustalać kolejkę priorytetów. O ile takie usprawnienie nie wpłynęłoby w znacznym stopniu na wydajność AZE to wydajność ANC mogłaby znacząco wzrosnąć. Możliwe są tu różne implementacje np.

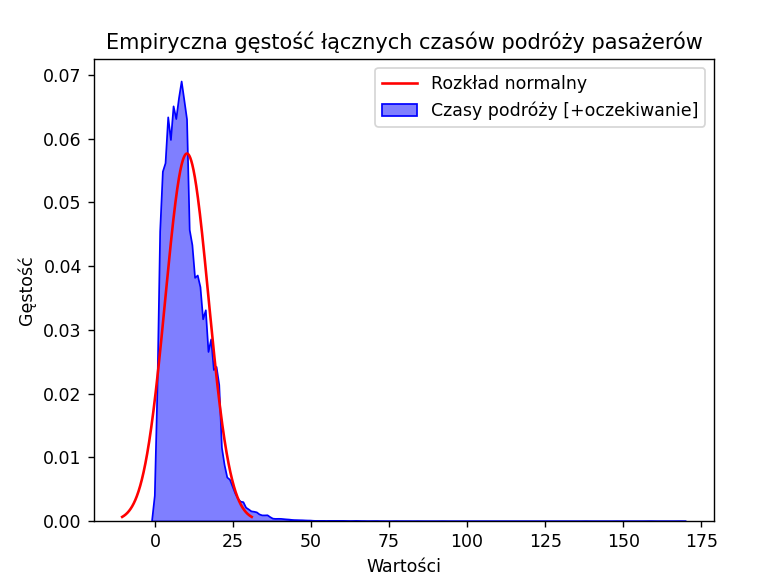
* ilość pasażerów na zgłoszenie – *waga zgłoszenia* wpływałaby, gdy dwa zgłoszenia miałyby tą samą fizyczną odległość od windy
* ilość pasażerów na zgłoszenie – *waga zgłoszenia* wpływałaby nawet jeśli zgłoszenie jest dalej, ale jego waga jest większa.

Złotym środkiem między tymi podejściami, w celu zbadania ich, wydaje się być zastosowanie drugiej opcji z parametrem odległości = 2; winda będzie priorytetyzowała zgłoszenia, nawet gdy będą dalej o 2. *Wartość 2 została wybrana na podstawie eksperymentów z różnymi parametrami i osiągnęła najlepsze wyniki.*

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Opis wygenerowany automatycznie

Rysunek 9: Eksperyment mający na celu wyznaczenie wartości głębokości wag ANC



Rysunek 10: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy w środowisku bloku mieszkalnego ANC zoptymalizowanego w zestawieniu z rozkładem normlanym

Rysunek 9. pokazuje rozkład 100.000 przebiegów ANC zoptymalizowanego o punkt postoju = 0 i priorytet wag (ilości pasażerów / zgłoszenie) w środowisku bloku mieszkalnego. Zgodnie z założeniami, przyniosło to poprawę, uzyskując najlepszy spośród algorytmów ANC średni łączny czas w tym środowisku symulacyjnym.

Tabela 6: Statystyki przebiegów zoptymalizowanego ANC dla 100.000 pasażerów w środowisku bloku mieszkalnego

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 100000 |
| Czas [ilość pięter]: | 799519 |
| Średni czas w windzie: | 5.97321 |
| Średni czas oczekiwania na windę: | 4.20865 |
| Średni czas podróży: | 10.1818 |
| Wariancja czasów podróży | 46.4019 |

Algorytm ANC po optymalizacji uzyskał średni czas podróży niższy o 0.445% od algorytmu optymalizowanego samym miejscem postoju oraz o 4.589% niższy niż czysty ANC.

**9.** **Wnioski dotyczące wydajności algorytmów AZE i ANC w środowiskach; losowym i bloku mieszkalnego.**

Tabela 7: Zestawienie średnich łącznych czasów wywołania algorytmów w różnych środowiskach

|  |  |  |
| --- | --- | --- |
| Algorytm | Środowisko symulacyjne | Średni łączny czas podróży |
| AZE | Środowisko losowe | 13.43996 |
| ANC | Środowisko losowe | 14.61532 |
| AZE + miejsce postoju | Blok mieszkalny | 9.87528 |
| ANC + m. p. + wagi | Blok mieszkalny | 10.1818 |
| ANC + miejsce postoju | Blok mieszkalny | 10.2273 |
| AZE | Blok mieszkalny | 10.41958 |
| ANC | Blok mieszkalny | 10.6713 |

Zarówno w środowisku losowym jak i symulującym 10-piętrowy blok mieszkalny o niskiej intensywności ruchu, algorytm AZE, spisywał się lepiej od algorytmu ANC. Był również bardziej podatny na optymalizacje.

Na uwagę zasługuje fakt, że optymalizacja jaką jest dodanie miejsca postoju zwiększyła ilość przejechanych pięter na 100.000 pasażerów o **0.655%** z wartości 799921 na 805162 w przypadku algorytmu AZE natomiast spadło **0.173%** z wartości 801933 na 800547. Więc w większości przypadków, gdzie intensywność ruchu jest stosunkowo niska i widać pewne wzorce wśród pasażerów, warto zastosować dodanie punktów postojowych, gdyż przy minimalnych zwiększeniach kosztów (lub ich obniżeniu w przypadku ANC) uzyskujemy znaczącą poprawę w efektywności (3 – 5%).

Samo przetestowanie algorytmów na dwóch środowiskach nie daje pełnego obrazu na ich zestawienie, niemniej wystarczy by postawić tezę mówiącą, że AZE może być lepszym rozwiązaniem, szczególnie gdy miejsce postoju będzie dynamicznie wyznaczane. Może to być dokonane poprzez sterownik analizujący wzorce lub sztuczną inteligencje wyuczoną w tym celu.

**10.** **Algorytmy uczenia maszynowego a optymalizacja łącznego czasu podróży**

W celu wyuczenia modelu sztucznej inteligencji do poruszania windą zostały wykorzystany kolejno metody uczenia przez wzmacnianie napisane przy użyciu *tensorflow* i *keras*:

* **Q-learning**: gdzie podczas uczenia windy do poruszania się po 10 (10+1) piętrach, powstał problem klątwy wymiarowości, wymagana była lepsza metoda bądź większe kompetencje.
* **Deep Q-Network**: model skutecznie nauczył się jeździć po 4 poziomowym budynku. Ale przez brak czasu, nie udało się uzyskać odpowiednika dla 10 piętrowego bloku.

**DQN**

Do budowy modelu DQN została użyta następująca struktura sieci neuronowej. Oraz strategia *epsilon greedy*, na 1000 epizodów, z buforem pamięci w postaci *Deque* o rozmiarze 4000 (batch = 32). Stopa dyskontowa = 0.9, aby zbalansować długo i krótkoterminowe nagrody. Optimizer to Adam a funkcja straty MSE co może nie być najlepszym wyborem w tym zagadnieniu.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Agent windy może wykonywać dwie akcje jazda w górę i dół. W celu uproszczenia otwieranie drzwi i ich zamykanie nie jest obsługiwane, a opcja postoju niekonieczna co wynika z architektury środowiska. Dodatkowo na stan składa się obserwowana rzeczywistość, opisana przez binarną przestrzeń 8 wartości w wektorze. Gdzie pierwsze 4 opisują, gdzie znajduje się aktualnie winda (tylko jeden z pierwszych 4 bitów może być = 1) oraz 4 kolejne przyjmujące wartości 1, gdy są na nich pasażerowie lub zadane piętro jest celem pasażerów w windzie.

Aby przetestować ten algorytm wykorzystana zostanie uproszczona do 4 pięter wersja środowiska w pełni losowego o średnim natężeniu ruchu.

Tabela 8: Statystyki przebiegów dla 10.000 DQN w środowisku średnio intensywnym 4-piętrowym.

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 10000 |
| Czas [ilość pięter]: | 22070 |
| Średni czas w windzie: | 2.6143 |
| Średni czas oczekiwania na windę: | 2.9714 |
| Średni czas podróży: | 5.5857 |
| Wariancja czasów podróży | 14.961 |

Obraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznie

Rysunek 11: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy DQN w środowisku średnio intensywnym 4-piętrowym w zestawieniu z rozkładem normlanym

Z uwagi na wprowadzenie nowego środowiska w celu analizy wyników DQN zostaną porównane z wynikami AZE w tym samym środowisku. Który osiągnął następujące wyniki:

**AZE**

**Obraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 12: Empiryczny rozkład gęstości łącznych czasów podróży pasażerów windy AZE w środowisku średnio intensywnym 4-piętrowym w zestawieniu z rozkładem normlanym

Tabela 9: Statystyki przebiegów dla 10.000 AZE w środowisku średnio intensywnym 4-piętrowym.

|  |  |
| --- | --- |
| Obsłużono pasażerów: | 10000 |
| Czas [ilość pięter]: | 22241 |
| Średni czas w windzie: | 2.0859 |
| Średni czas oczekiwania na windę: | 1.6099 |
| Średni czas podróży: | 3.6958 |
| Wariancja czasów podróży | 3.12626 |

Wyniki te pokazują, że choć DQN działa, wymaga on poprawy, gdyż jego wyniki są znacznie gorsze niż algorytmu AZE. Wskazuje to, że wyuczony model DQN wymaga dostrojenia hiperparametrów i przede wszystkim większej ilości epizodów do nauki, lub wykorzystanie innej architektury takiej jak *Double DQN*, czy też usprawnienie wektora stanów.

**11.** **Wnioski ogólne**

Sprawa optymalizacji *czasu średniego przejazdu i oczekiwania* na windę nie jest sprawą tak oczywistą jak mogłoby się wydawać. Symulacje w różnych środowiskach pokazały wady i zalety analizowanych algorytmów oraz potencjalne sposoby na ich usprawnienie. Podstawowymi usprawnieniami, które należałoby rozpatrzyć projektując „idealny*”* sterownik windy są m.in.:

* **Dodanie punktów postoju windy**: Rozwiązanie to przede wszystkim sprawdza się w blokach mieszkalnych bądź generalizując, budynkach, gdzie można znaleźć obserwowalne wzorce wśród podróży (start i cel) pasażerów. W tym celu można zastosować np. sztuczną inteligencje do identyfikacji takich wzorców. Można też zastosować statycznie ustawiony punkt postoju, który według symulacji zawartych w sprawozdaniu, może przynieść **3-5%** poprawę łącznego czasu podróży.
* **Dodanie liczników pasażerów na zgłoszenie**: Docelowo, znając ilość pasażerów, algorytm może przypisywać priorytet bazując w pewnej proporcji na ilości pasażerów w zgłoszeniu. Wymagałoby to jednak instalacji dodatkowego oprogramowania do struktury dźwigów osobowych.
* **Wykorzystanie AI do ustalania kolejki obsługi zgłoszeń**: Lepszy niż omawiany algorytm DQN, mógłby mieć potencjał osiągnąć niższy łączny średni czas podróży, niż AZE.
* **Zwiększanie prędkości przejazdu dla odległych zgłoszeń**: Manipulacja maksymalną prędkością jest oczywistym i prostym usprawnieniem, gdy chodzi o czas przejazdu. Szczególnie, gdy winda jest pusta i jedzie po pasażerów, mogłaby osiągać większe prędkości.
* **Dodanie drugiego dźwigu**: Wykonywanie symulacji i zbieranie statystyk, podobnych do tych z tego sprawozdania może pomóc w wykryciu anomalii pracy windy w tym potrzeby instalacji drugiego dźwigu by zwiększyć wydajność systemu.

Warto zaznaczyć, że pomimo ignorowania aspektów związanych z energią, symulacje pokazały, że optymalizacja czasów podróży nie zawsze musi iść naprzeciw wydajności energetycznej. Więc dodatkowym wnioskiem jest fakt, że: *optymalizowanie wydajności algorytmu pod względem czasów przejazdu się opłaca*. Proporcja wzrostu kosztów do wzrostu wydajności czasowej można szacować na około 1/5.

W zestawieniach, bazujących na symulowanych środowiskach, najlepszym algorytmem okazał się być, zgodnie z założeniem, *algorytm zgodnych ekstremów*. Jego skuteczność potwierdzona została brakiem odchyleń na empirycznych rozkładach jego przebiegów.

Kody do algorytmów użytych w sprawozdaniu znajdują się w **publicznym repozytorium**: *https://github.com/Afraiven/Elevator*