Symulator inteligentnego sterowania windą
Sprawozdanie

Artur Gajowniczek
Michał Kośmider
Michał Kalisiak
Karol Mąkosa
Rafał Woźniak

14 czerwca 2018

Spis treści

1 Wprowadzenie **2**

2 Istniejące rozwiązania **2**

3 Opis architektury programu **3**
3.1 Moduł interfejsu graficznego 3
3.2 Moduł kontroli windy 4
3.3 Moduł zegara 5
3.4 Moduł generacji ludzi 5
3.5 Moduł algorytmów sterowania windą 5
3.6 Moduł statystyk 5

4 Testowanie statyczne **6**

5 Testowanie dynamiczne **8**

6 Bibliografia **12**

1 Wprowadzenie

Systemy wbudowane i czasu rzeczywistego znajdują w dzisiejszym świecie coraz więcej zastosowań. Nowoczesna technologia przyzwyczaiła ludzi do bardzo krótkiego czasu odpowiedzi w niemalże wszystkich systemach komputerowych. Powstało też wiele systemów, w których z góry określony czas odpowiedzi jest kluczowy dla ich poprawnego działania i to właśnie one są nazywane systemami czasu rzeczywistego. Systemy wbudowane natomiast są nierozerwalnie połączone z urządzeniem, które obsługują. Jednym z przykładów systemu, który jest jednocześnie systemem wbudowanym oraz czasu rzeczywistego, jest system inteligentnego sterowania windą.

Problem, który rozwiązuje system sterowania windą jest w dużym stopniu podobny do dobrze znanego Problemu Komiwojażera, jednak występuje kilka znaczących różnic między nimi. Główną różnicą jest fakt, że w Problemie Komiwojażera całe zadanie jest od początku znane, natomiast w naturalnym środowisku windy warunki zadania cały czas się zmieniają, ponieważ pojawiają się nowi ludzie. Z tego powodu nie można z góry wyliczyć optymalnego rozwiązania, algorytm sterowania musi brać pod uwagę zmiany przy podejmowaniu kolejnych decyzji.

2 Istniejące rozwiązania

Inteligentne sterowanie windą nie jest nowym problemem i został już opracowany niejeden algorytm z nim związany. Większość z nich zajmuje się sterowaniem grupą wind, a nie pojedynczą windą. Poniżej znajduję się lista kilku z tych algorytmów oraz strategii wraz z krótkimi opisami.

- Strategia kontroli kolektywnej [1] – w tym algorytmie winda porusza się w jednym kierunku zabierając tylko pasażerów jadących w tym samym kierunku. Gdy wszyscy pasażerowie wysiądą i nie ma następnych pasażerów znajdujących się dalej w danym kierunku niż winda, winda zmienia kierunek jeśli są jakieś inne wezwania. W przeciwnym wypadku winda będzie czekać na piętrze na którym wysiadł ostatni pasażer.
- Strategie przeszukujące [1] – w odróżnieniu od poprzedniej strategii, która należy do rodziny algorytmów zachłannych, strategie przeszukujące będą analizować wiele (lub wszystkie) możliwości i ich konsekwencje, a następnie podejmować decyzje wybierając spośród możliwości tę, która najlepiej spełnia zadane kryterium optymalizacyjne np. średni czas oczekiwania, średni czas obsłużenia, itp.. Strategie przeszukujące potrzebują więcej czasu na podjęcie decyzji, co może negatywnie wpływać na na średni czas oczekiwania, ale całościowo wyniki będą lepsze.

- Strategia oparta na regułach [1] – jest to strategia oparta na weryfikacji logicznych zdań typu „JEŚLI warunek TO skutek". Takie reguły są tworzone na podstawie wiedzy ekspertów oraz przeprowadzanych badań.
- Algorytm genetyczny [1] – używając algorytmu genetycznego system sterujący windą, może „nauczyć się" reagować na różne sytuacje poprzez ustawienie funkcji przystosowania zachowania jako funkcji optymalizowanego parametru np. średniego czasu oczekiwania.
- Model zużycia energii [2] – jest to model oparty o optymalizację przejechanych pięter do rozwiezienia wszystkich pasażerów w danej chwili czasowej.

3 Opis architektury programu

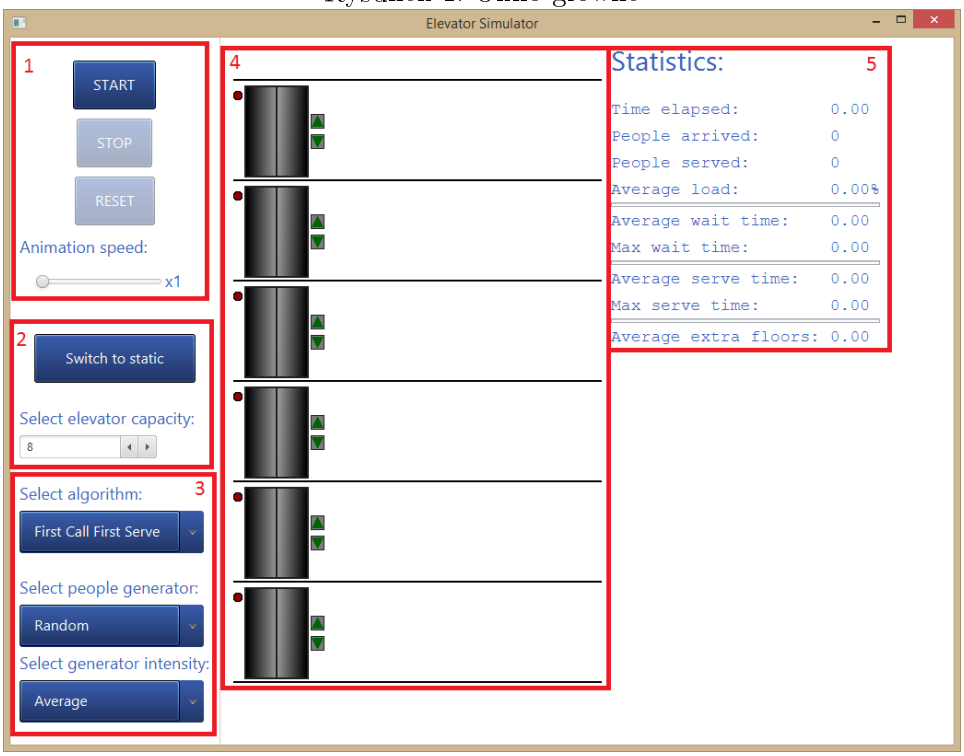
Symulator inteligentnego sterowania windą w naszym projekcie został zaimplementowany przy użyciu języka **Java** w wersji 8. Dodatkowo użyta została biblioteka do interfejsu graficznego użytkownika – **JavaFx**.

Działanie programu można podzielić na kilka niezależnych, lecz połączonych interfejsami modułów.

3.1 Moduł interfejsu graficznego

Użycie biblioteki **JavaFx** pozwoliło na uzyskanie ładnego okna graficznego, które prezentuje przejrzysty interfejs.

Rysunek 1: Okno główne



1. Przyciski sterujące rozpoczęciem i zakończeniem pracy windy, zresetowaniem windy do stanu początkowego oraz suwak służący do przyspieszania animacji

2. Przyciski zmieniające tryb pracy generatora ludzi (statyczny i dynamiczny) oraz ustawiający maksymalną pojemność windy

3. Rozsuwane listy wyboru algorytmu sterowania windą oraz wyboru generatora ludzi i jego intensywności generacji (tylko w przypadku trybu dynamicznego)

4. Obszar prezentujący aktualny stan windy

5. Obszar zawierający statystyki

3.2 Moduł kontroli windy

Zajmuje się przechowywaniem oraz modyfikacją aktualnego stanu windy za pomocą maszyny stanów. Przechowuje on także informacje o ludziach znajdujących się na poszczególnych piętrach oraz w windzie. Generuje zdarzenia, które dopisywane są do kolejki zdarzeń w module zegara.

3.3 Moduł zegara

Zarządza zdarzeniami z poszczególnych modułów, które dzięki działaniu zegara, odbywają się w predefiniowanych chwilach czasowych. Każde zdarzenie w systemie zajmuje określoną liczbę tyknięć zegara, co symuluje upływ czasu (na przykład przejechanie piętra trwa dłużej niż otworzenie drzwi).

3.4 Moduł generacji ludzi

Pozwala na generowanie ludzi chcących skorzystać z windy na różne sposoby. Każdy z algorytmów bazuje na rozkładzie Poissona, którego parametr lambda można modyfikować przy użyciu interfejsu graficznego (poprzez zmianę intensywności generatora). Zaimplementowane metody generowania ludzi to:

- Random – algorytm losowy, żadne piętro nie jest faworyzowane w tym algorytmie, ludzie pojawiają się na wszystkich piętrach z takim samym prawdopodobieństwem i chcą jechać na losowe, inne piętra
- Office Morning – symuluje napływ ludzi do biurowca w godzinach porannych. Najwięcej ludzi pojawia się na parterze, natomiast na innych piętrach ludzie pojawiają się sporadycznie
- Office Evening – symuluje wychodzenie ludzi z biurowca w godzinach popołudniowo-wieczornych. Ludzie na parterze pojawiają się sporadycznie, natomiast na pozostałych piętrach ludzie pojawiają się z takim samym prawdopodobieństwem i wszyscy chcą jechać na parter

3.5 Moduł algorytmów sterowania windą

Pozwala na wybór algorytmu służącego do podejmowania decyzji przez moduł kontroli windą. Zaimplementowane algorytmy to:

- First Call First Serve – obsługuje pierwsze zgłoszone żądanie, z wyjątkiem sytuacji, gdy po drodze jest w stanie obsłużyć jakieś inne żądanie
- Momentum – realizuje strategię kontroli kolektywnej
- Morning – stara się zoptymalizować czas oczekiwania w biurowcu w godzinach porannych, poprzez zjeżdżanie na parter w przypadku bezczynności windy

3.6 Moduł statystyk

Zajmuje się wyliczaniem statystyk widocznych w obszarze statystyk. Są to (każda statystyka określająca czas jest wyrażona w tyknięciach zegara):

- Elapsed time – czas, który upłynął od rozpoczęcia symulacji

- People arrived – liczba ludzi, która pojawiła się w sumie na wszystkich piętrach od rozpoczęcia symulacji
- People served – liczba ludzi, która została obsłużona od rozpoczęcia symulacji
- Average load – średnie obciążenie windy wyrażone jako procent jej pojemości od rozpoczęcia symulacji
- Average wait time – średni czas oczekiwania ludzi zanim mogli wsiąść do windy
- Max wait time – najdłuższy czas oczekiwania osoby zanim mogła wsiąść do windy
- Average serve time – średni czas obsłużenia ludzi (od pojawienia się na piętrze początkowym, do dotarcia na piętro docelowe)
- Max serve time – najdłuższy czas obsłużenia osoby
- Average extra floors – średnia liczba pięter przejechanych ponad najmniejszą możliwą liczbę pięter dla każdego pasażera

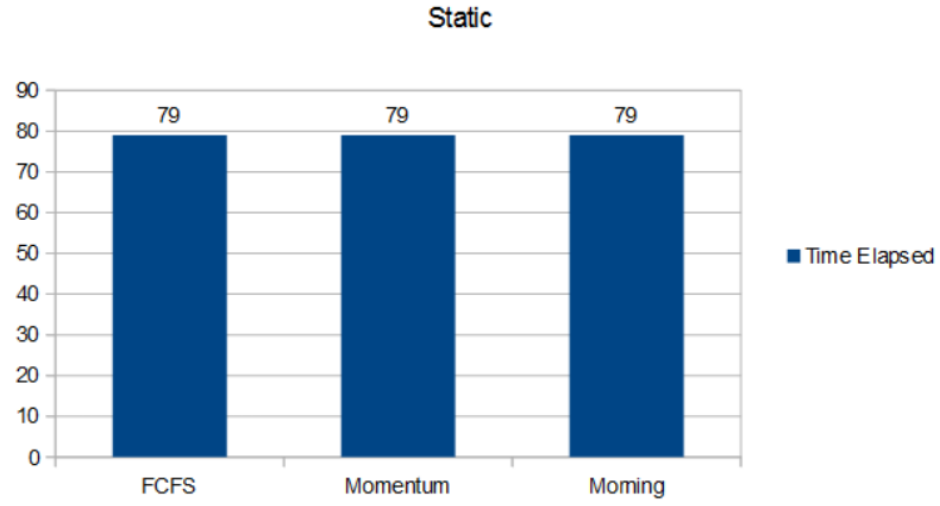
4 Testowanie statyczne

Zaczęliśmy od przetestowania algorytmów w wersji statycznej, to znaczy ludzie byli wygenerowani na samym początku symulacji, a w później generator był wyłączony i nowi ludzie nie pojawiali się. Winda zaczynała zawsze z parteru. Rozkład ludzi na którym testowaliśmy algorytmy to:

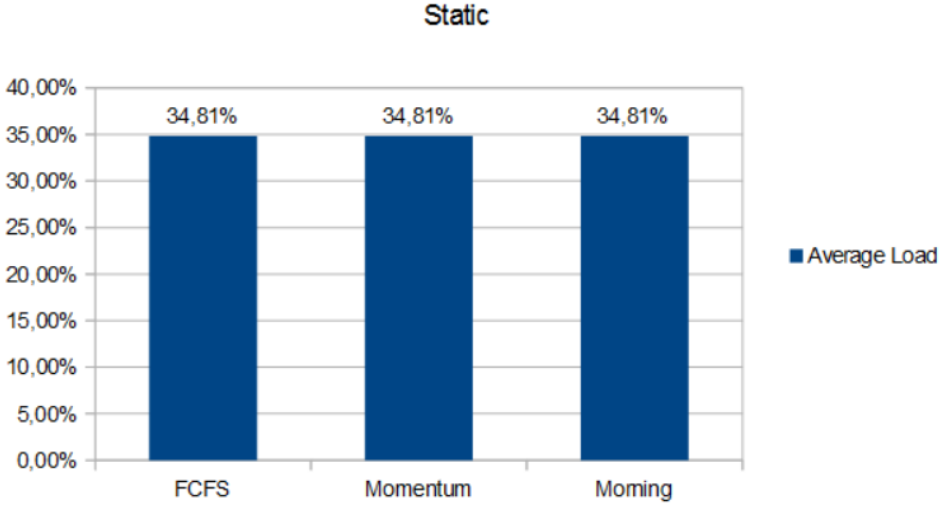
- 1 osoba na parterze chcąca jechać na piąte piętro
- 2 osoby na pierwszym piętrze, z których jedna chce jechać na trzecie piętro, a druga na parter
- 6 osób na trzecim pierze, z których jedna chce jechać na pierwsze piętro, a pozostałe na piąte piętro

Testy polegały na uruchomieniu każdego algorytmu w statycznym trybie symulatora i zatrzymaniu symulacji gdy ostatnia osoba wysiadła z windy i spisaniu statystyk. Wyniki testów widać na poniższych wykresach.

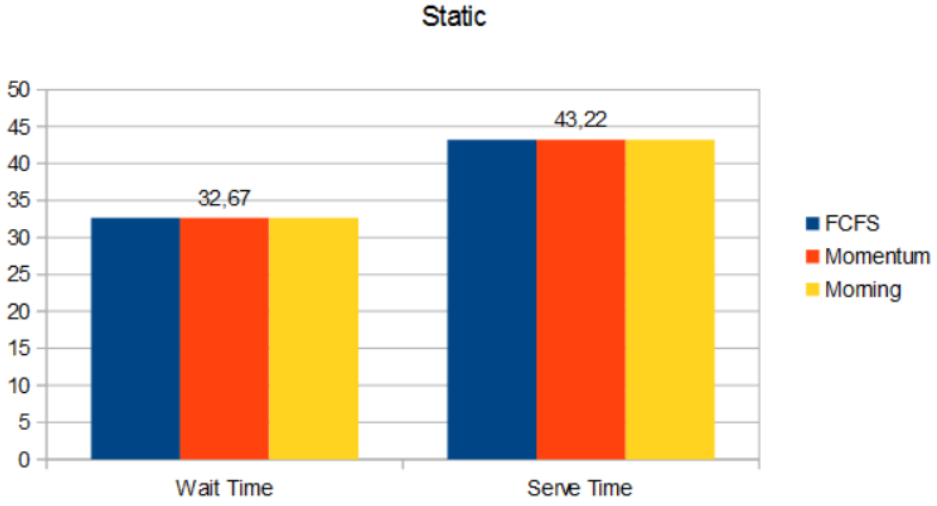
Rysunek 2: Czas, który upłynął od rozpoczęcia symulacji



Rysunek 3: Średnie obciążenie windy



Rysunek 4: Średni czas czekania oraz obsłużenia

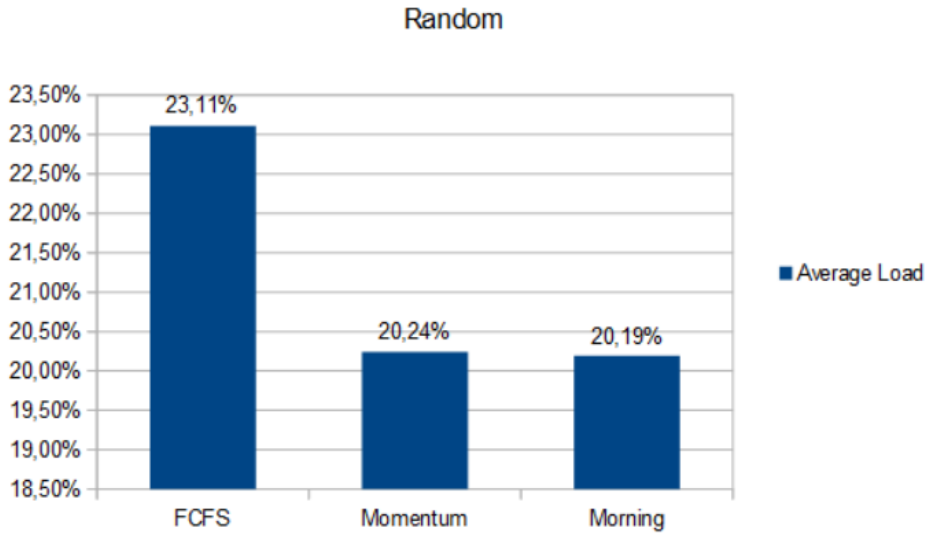


Jak widać w przypadku statycznym każdy algorytm osiągnął te same wyniki.

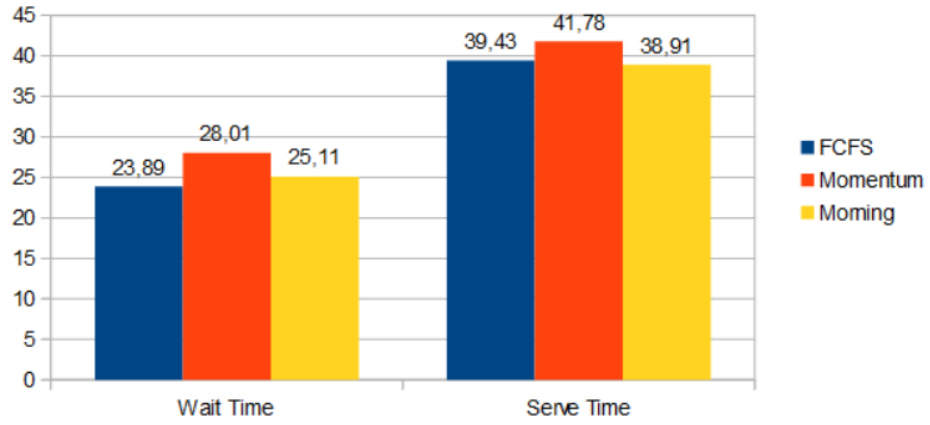
5 Testowanie dynamiczne

Następnie testowaliśmy algorytmy z różnymi generatorami w dynamicznym trybie symulatora. W chwili startu na piętrach nie czekali żadni ludzie, byli oni generowani losowo w oparciu o wybrany generator. Ponieważ generator nie przestaje działać sam z siebie, w celu porównania algorytmów zatrzymywaliśmy symulację po stałym czasie (wynoszącym 1000 tyknięć zegara) i spisywaliśmy statystyki. Wyniki testów widać na poniższych wykresach.

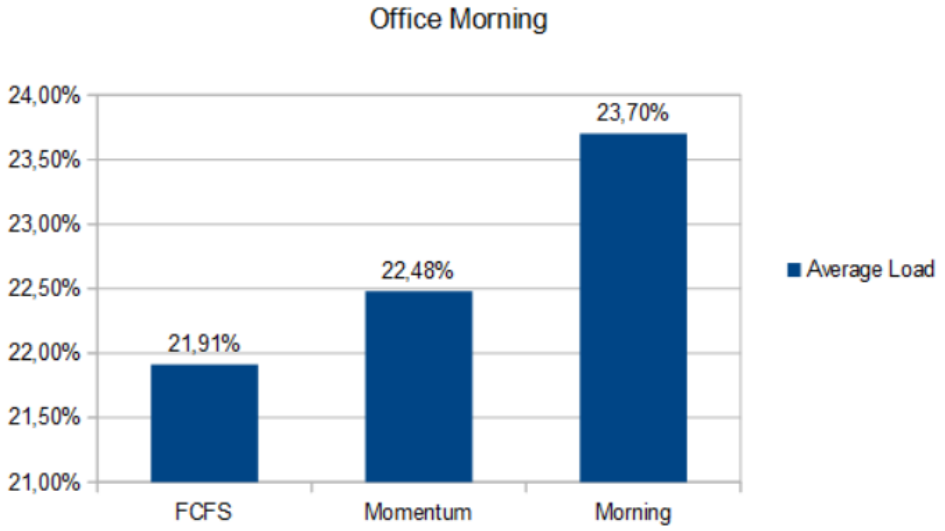
Rysunek 5: Generator losowy - średnie obciążenie windy



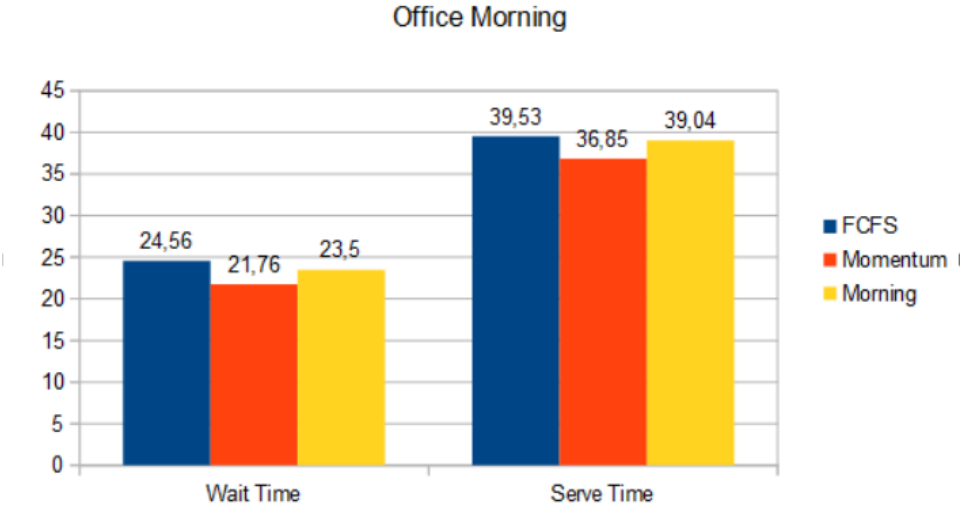
Rysunek 6: Generator losowy - średni czas czekania oraz obsłużenia



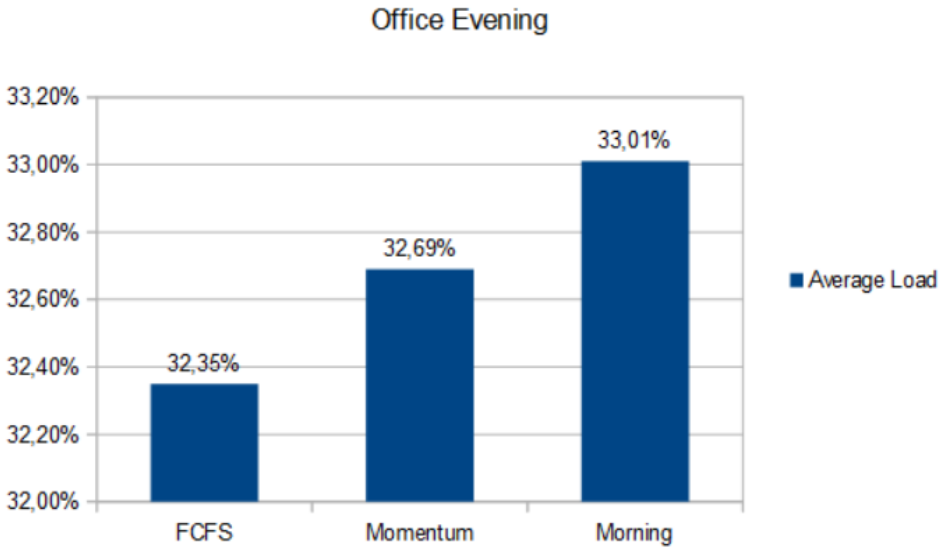
Rysunek 7: Generator poranny - średnie obciążenie windy



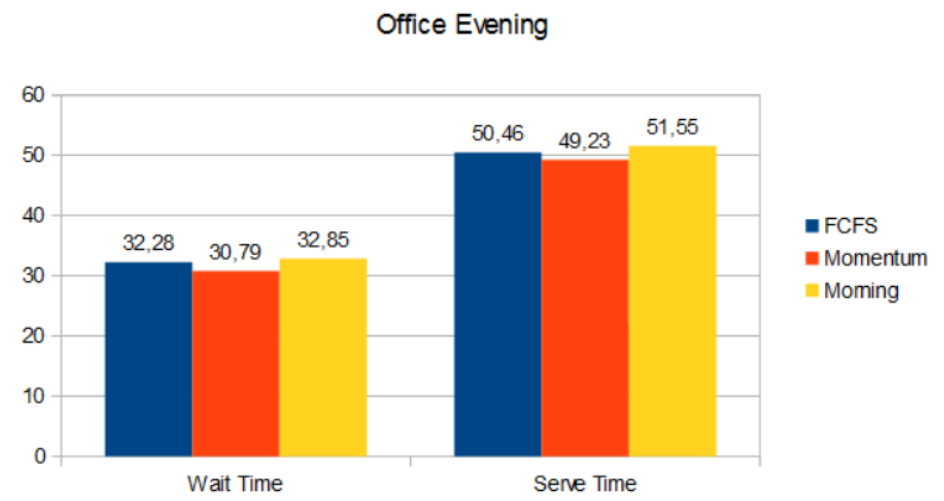
Rysunek 8: Generator poranny - średni czas czekania oraz obsłużenia



Rysunek 9: Generator wieczorny - średnie obciążenie windy



Rysunek 10: Generator wieczorny - średni czas czekania oraz obsłużenia



Widać tu nieco zaskakujące wyniki, ponieważ w przypadku generatora porannego algorytm poranny nie osiąga najlepszych wyników (poza średnim obciążeniem), ale może być to spowodowane przypadkową generacją ludzi faworyzującą inne algorymy.

6 Bibliografia

[1] https://www.diva-portal.org/smash/get/diva2:668654/FULLTEXT01.pdf
[2] http://www.csc.kth.se/utbildning/kth/kurser/DD143X/dkand13/Group3Johan/report/alexandra.nordin.frederick.ceder.report.pdf