Inteligencja Obliczeniowa 2024

Optymalizacja ruchu pojazdów w mieście za pomocą narzędzia MatSIM

Autorzy:
Antonina Kuś
Szymon Żychowicz

1. Etap: Analiza problemu i dziedziny

Optimization and simulation of fixed-time traffic signal control in real-world applications.

https://www.sciencedirect.com/science/article/pii/S1877050919305770?ref=pdf_download&fr=RR-2&rr=868f0222797527b4

Evolutionary design optimization of traffic signals applied to Quito city.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5728506/

Modeling crossroads in MATSim The case of traffic-signaled intersections.

https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/429954/1/ab1546.pdf

2. Etap: analiza i wybór narzędzi

W zadaniu będziemy korzystać z programu MatSim, który służy do symulacji transportu. Aby kontrolować sygnalizację świetlną, użyjemy biblioteki Signals 11.0. Ta biblioteka umożliwia konfigurowanie cykli świateł za pomocą plików XML zawierających ustawienia programu.

https://matsim.org/apidocs/signals/11.0/

3. Etap: określone cele i zakres prac, uruchomione narzędzia

Celem projektu jest optymalizacja ruchu pojazdów za pomocą dostosowania cykli świateł drogowych. Optymalizacja przebiegać będzie przy użyciu algorytmów ewolucyjnych. W naszym problemie osobnikiem będzie cykl świateł, który ulegał będzie zmianom na podstawie uśrednionego czasu oczekiwania samochodów w symulacji. Wykorzystany algorytm optymalizacji będzie algorytmem jednokryterialnym.

Wybraliśmy bibliotekę PyMoo ze względu na łatwość implementacji, jaką oferuje język Python. Znajduje się tam wiele algorytmów, z których wybraliśmy 3, które mogą pasować do naszego problemu. Są to:

1. Genetic Algorithm - jest to podstawowa implementacja algorytmu genetycznego. Może być łatwo dostosowywana i stosowana do szerokiej kategorii problemów.

2. Evolutionary Strategy - stosowany jest do rozwiązywania problemów optymalizacji wartości rzeczywistych.

3. Improved Stochastic Ranking Evolutionary Strategy - udoskonalona wersja algorytmu SRES (strategia ewolucyjna z obsługą ograniczeń przy użyciu stochastycznego rankingowania) zdolna efektywnie radzić sobie z zależnymi zmiennymi.

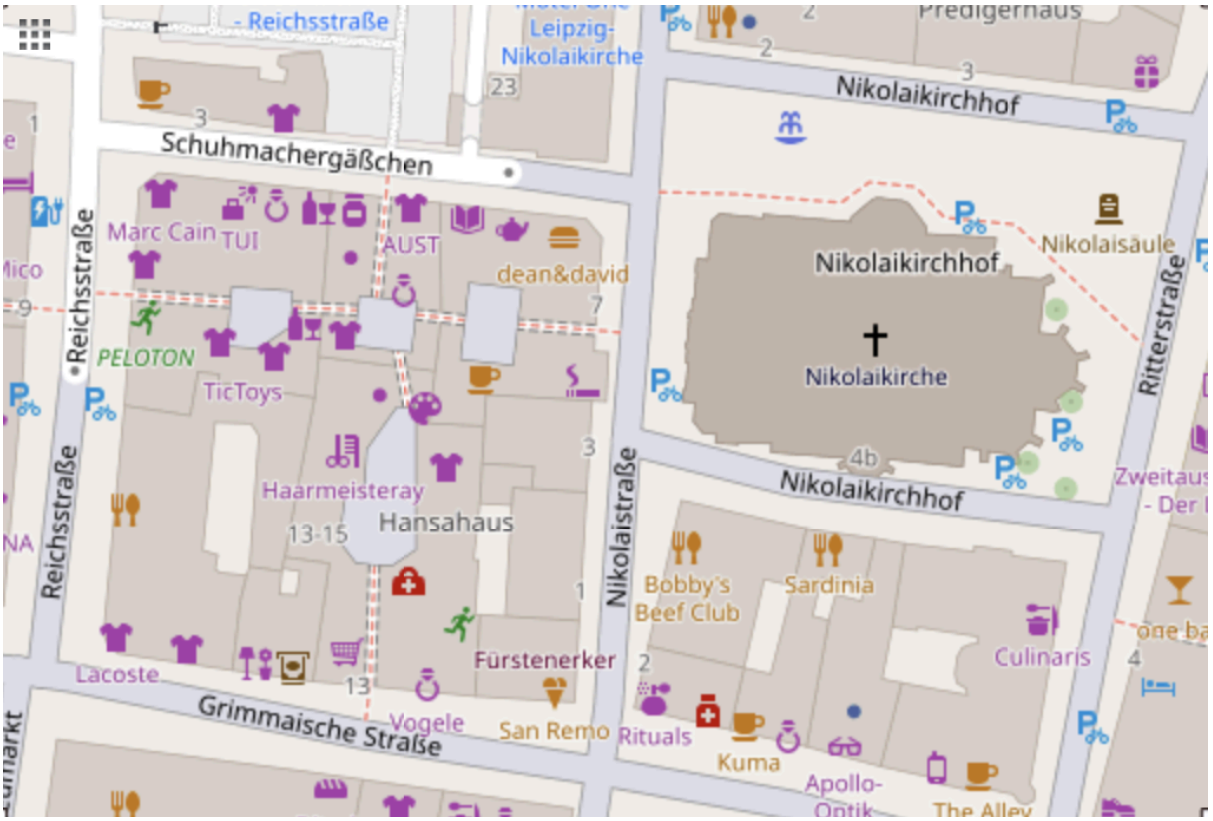
Przykładowy osobnik - cykl świateł zapisany w pliku .xml:

```
<signalPlan id="1">
    <start daytime="17:50:00" />
    <cycleTime sec="90"/>
    <offset sec="0"/>
    <signalGroupSettings refId="4">
        <onset sec="0"/>
        <dropping sec="10"/>
    </signalGroupSettings>
</signalPlan>
```

Do programu PyMoo użyta będzie tablica zawierająca kolejno sekundy cyklu, kiedy zapala się zielone światło oraz sekundy cyklu, kiedy zapala się czerwone światło.

```
[onset1,onset2,...,onsetn,dropping1,dropping2,...,droppingn]
```

W przypadku uruchomienia programu na bardzo niewielkim wycinku Lipska otrzymujemy OutOfMemoryError (na udostępnieniu IDE 24GB RAM):



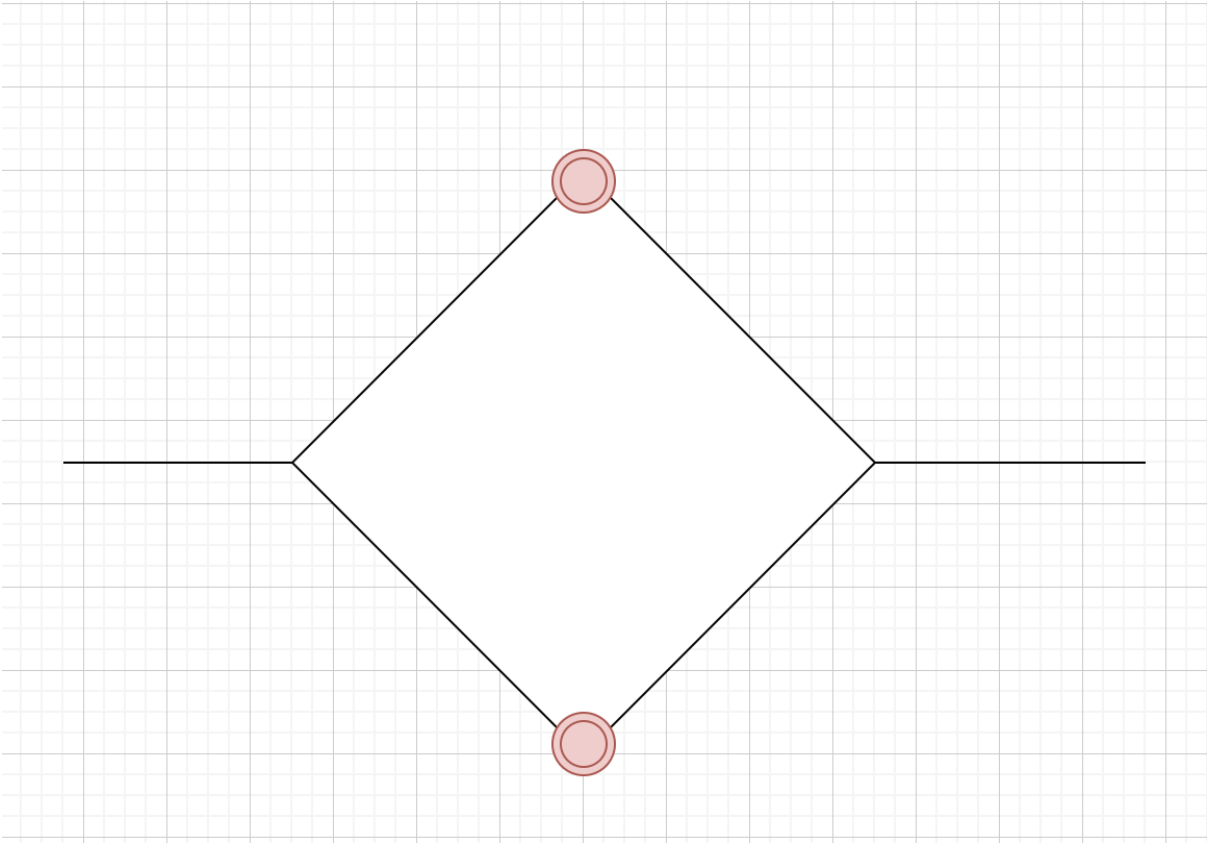
Rysunek 1: Fragment Lipska, dla którego występuje OutOfMemoryError

```
2024-04-14T19:57:18,645  INFO Counter:75 [PopulationWriter] dumped person # 115365
2024-04-14T19:57:19,685  INFO MemoryObserver:42 used RAM: 5717 MB  free: 426 MB  total: 6144 MB
2024-04-14T19:36:28,626  INFO ChooseRandomLegModeForSubtour:137 Chain based modes: [car, bike]
2024-04-14T19:36:28,626  INFO ChooseRandomLegModeForSubtour:137 Chain based modes: [car, bike]
2024-04-14T19:36:28,626  INFO ChooseRandomLegModeForSubtour:137 Chain based modes: [car, bike]
2024-04-14T19:36:37,820 ERROR MatsimRuntimeModifications:76 Getting uncaught Exception in Thread main
com.google.inject.ProvisionException: Unable to provision, see the following errors:

1) [Guice/ErrorInjectingConstructor]: OutOfMemoryError: Java heap space
  at TripRouter.<init>(TripRouter.java:102)
  while locating TripRouter

Learn more:
  https://github.com/google/guice/wiki/ERROR_INJECTING_CONSTRUCTOR

1 error

======================
Full classname legend:
======================
TripRouter:       "org.matsim.core.router.TripRouter"
========================
End of classname legend:
========================
```

Rysunek 2: OutOfMemoryError przy próbie uruchomienia programu dla 24GB RAM

4. Etap: prototyp, pierwsze działające elementy

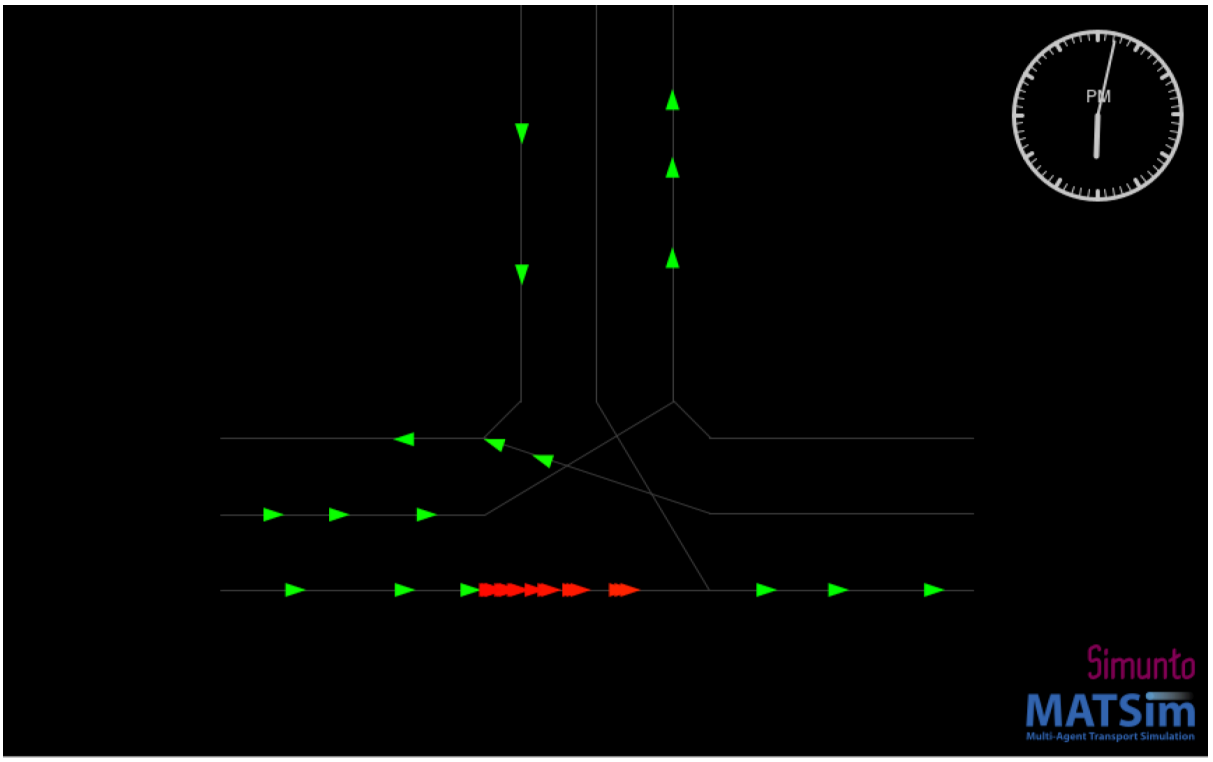
W ramach kolejnego etapu, udało nam się uruchomić symulację na wygenerowanym przez nas skrzyżowaniu i zaobserwować działanie świateł w programie VIA. Dodatkowo stworzyliśmy kod, za pomocą którego odbywać będzie się finalna symulacja. Po zagłębieniu się w problem, który pojawił się w etapie 3, udało nam się ograniczyć czas wykonywania się symulacji do około minuty, natomiast dzieje się to na zaprojektowanym przez nas ręcznie pliku .xml, a nie na danych pobranych z fragmentu miasta. Poniżej znajduje się schemat drogi, którego użyliśmy do symulacji. Na czerwono zaznaczono miejsca, w których umieszczone zostały światła.



Rysunek 3: Zaprojektowany fragment drogi wraz ze światłami.

5. Etap: prototyp, działa większość funkcji

Prawie wszystkie pojedyńcze funkcjonalności zostały zaimplementowane i działają w zamierzony sposób. Dodatkowo, udało się przeprowadzić symulację na wybranym skrzyżowaniu z dobranymi działającymi czasami świateł, które następnie będą mogły być modyfikowane przed algorytm. Udał się także stworzyć program zarówno do odczytu wartości z pliku, jak i zapisywania do .xml nowych ustawień świateł.



Rysunek 4: Wizualizacja skrzyżowania w programie VIA.