

Tecnológico Nacional de México
Instituto Tecnológico de Tijuana

Doctorado en Ciencias de la Ingeniería

**Interactive system for ambulance dispatching in the city of Tijuana:
Simulator**

By:

M. C. Noelia Araceli Torres Cortés

Thesis Director:

Dra. Yazmin Maldonado Robles
Dr. Leonardo Trujillo Reyes

February 24, 2022



TECNICOLOGICO
NACIONAL DE MEXICO



Content

- ① Introduction
- ② Problem statement
- ③ Objective
- ④ An overview of the simulation module
 - 4.1 Requirements to run the simulator module
- ⑤ Simulator module (original version from UCSD)
- ⑥ Integration of the simulator module to the interactive system for ambulance dispatching in the city of Tijuana
 - 6.1 Updated simulator module
- ⑦ Conclusion
- ⑧ Outlook

1. Introduction

Emergency medical system

In recent decades, a major problem has been to reduce the response times, a critical factor because when the service is not provided quickly it can put people's lives at risk.

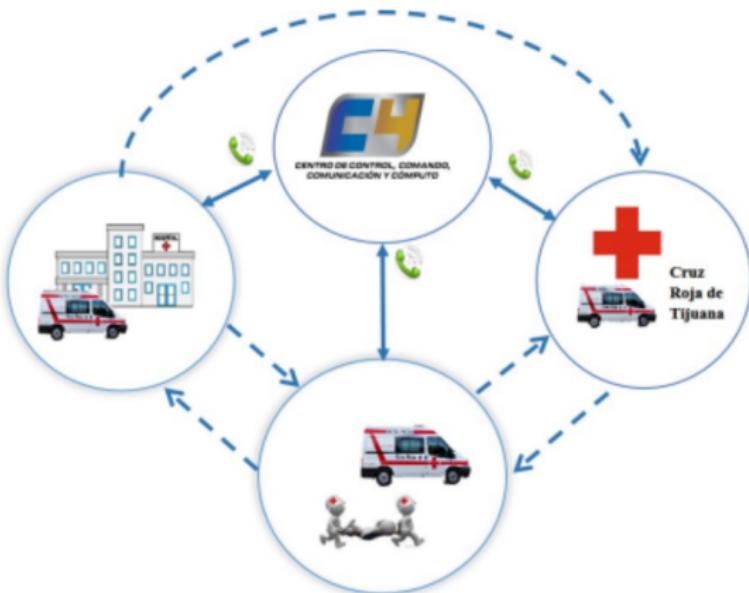


Figure 1: Emergency medical service system by the Tijuana Red Cross.

2. Problem statement

Problem statement

This work aims to address the problem of dispatching prehospital emergency units (ambulances) in the city of Tijuana, considering the travel time (duration of the EMS) and relocation of the units.

According to the American Heart Association an EMS must be attended between 4 and 6 minutes [1]. However, given the geographical conditions of the city and the few resources that the Tijuana Red Cross has it is difficult to attend this expectation (in Tijuana an EMS is attended in approximately 14 minutes).

Other important factors are:

- Traffic conditions (peak hours, weekdays, etc.)
- The priority of the call
- Streets and avenues in poor condition
- Hard to find the geographical location of the call

[1] American Heart Association AHA.<http://circ.ahajournals.org/content/84/2/960.citation.2018>

Problem statement

This work aims to address the problem of dispatching prehospital emergency units (ambulances) in the city of Tijuana, considering the travel time (duration of the EMS) and relocation of the units.

According to the American Heart Association an EMS must be attended between 4 and 6 minutes [1]. However, given the geographical conditions of the city and the few resources that the Tijuana Red Cross has it is difficult to attend this expectation (in Tijuana an EMS is attended in approximately 14 minutes).

Other important factors are:

- Traffic conditions (peak hours, weekdays, etc.)
- The priority of the call
- Streets and avenues in poor condition
- Hard to find the geographical location of the call

[1] American Heart Association AHA.<http://circ.ahajournals.org/content/84/2/960.citation.2018>

Problem statement

Tijuana features:

- Approximately 1.6 million inhabitants.

RCT features:

- Covers approximately 98% of all requests for EMS
- With approximately between 12 to 16 ambulances.
- 6 static bases.



Figure 2: Tijuana, Baja California.

3. Objective

Objective

The objective is to generate an optimal dispatch rule of the ambulances for the city of Tijuana, that minimizes the average response time of the ambulances during a given period of time, but at the same time, maximizes the average coverage provided during that period of time.

4. Simulator Module Overview

Simulator module

The simulator is based on state machines that allow to evaluate different events, cases (emergency calls), ambulance location, ambulance dispatch.

- The original version (UCSD) of the simulator is available at:

<https://github.com/EMSTrack/Algorithms>

- The modified version of the simulator is available at:

<https://onx.la/75e83>

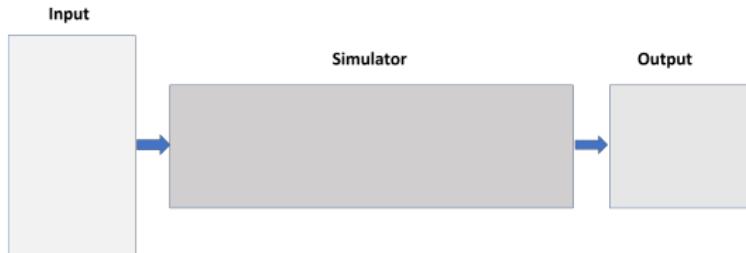


Figure 3: Simulator module.

5.1 Requirements to run the simulator module

A horizontal progress bar consisting of a thick dark grey segment followed by a thinner white segment.

Software requirements

Software recommendations

System Operative UNIX-based operating system such as Mac OS or Linux.

Integrated development environment PyCharm

Python Version 3.7

Table 1: Software recommendations

Setting up a Python interpreter in PyCharm

Open PyCharm and find the **Algorithms** folder that contains the project.

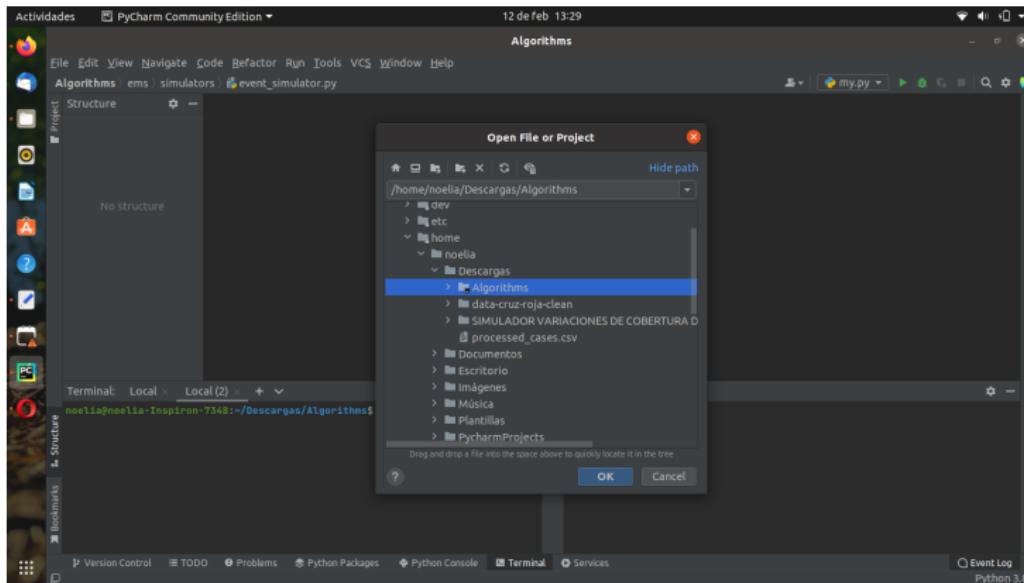


Figure 4: Initial PyCharm setup.

Setting up a Python interpreter in PyCharm

In PyCharm select **File – > Settings**, in this window we will have to include the interpreter with which we want to work in the project.

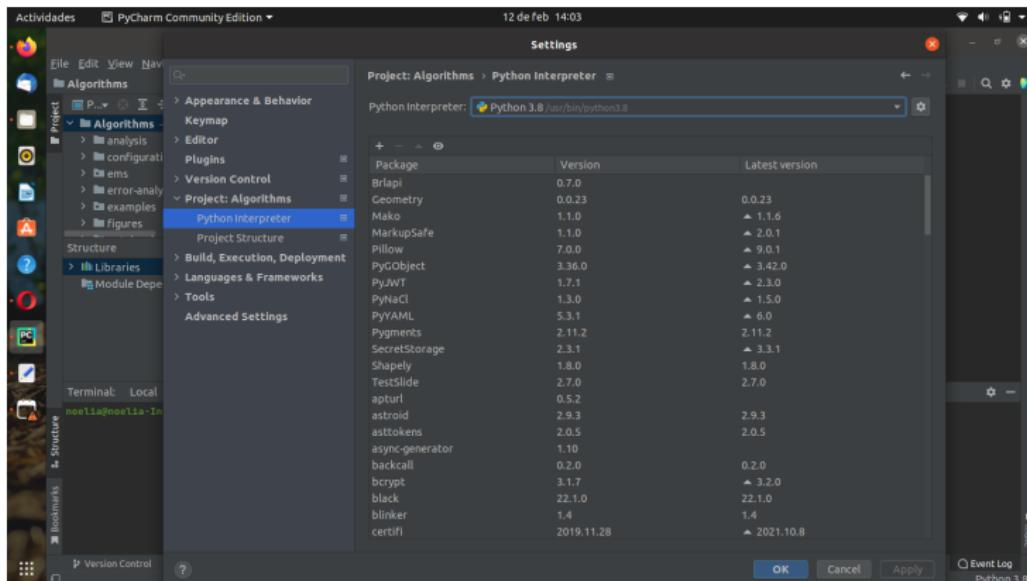


Figure 4: Initial PyCharm setup.

Installing Python packages

To download and install the packages required for the Python project. A text file (called requirements.txt) is used where the packages are located for automatic installation.

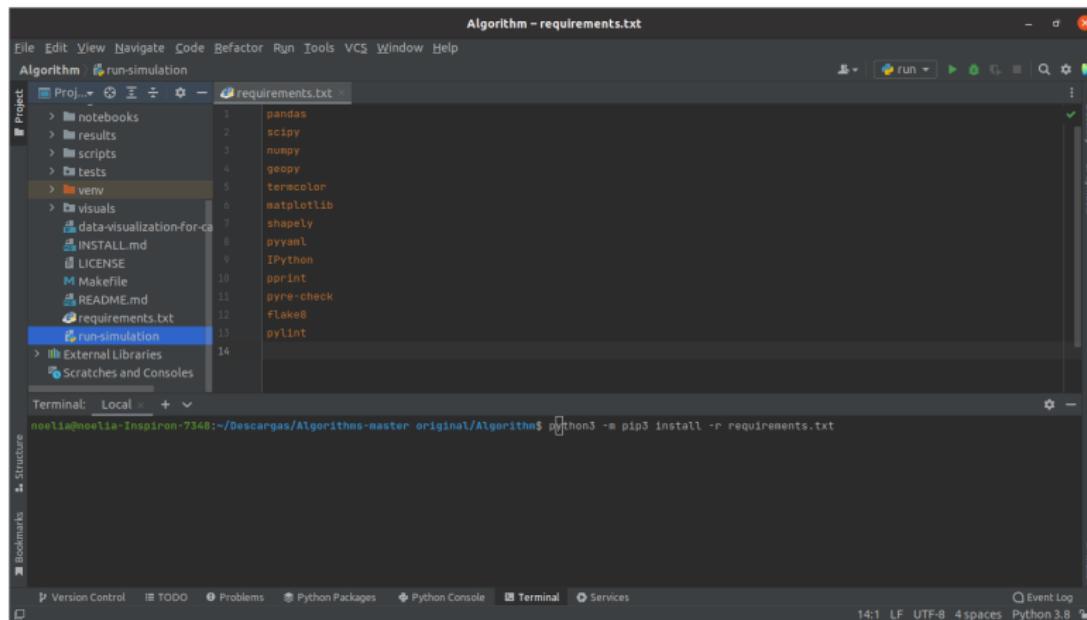


Figure 5: Installing Python packages.

Installing Python packages

Simply run the following command in the terminal:

```
python3 -m pip3 install -r requirements.txt
```

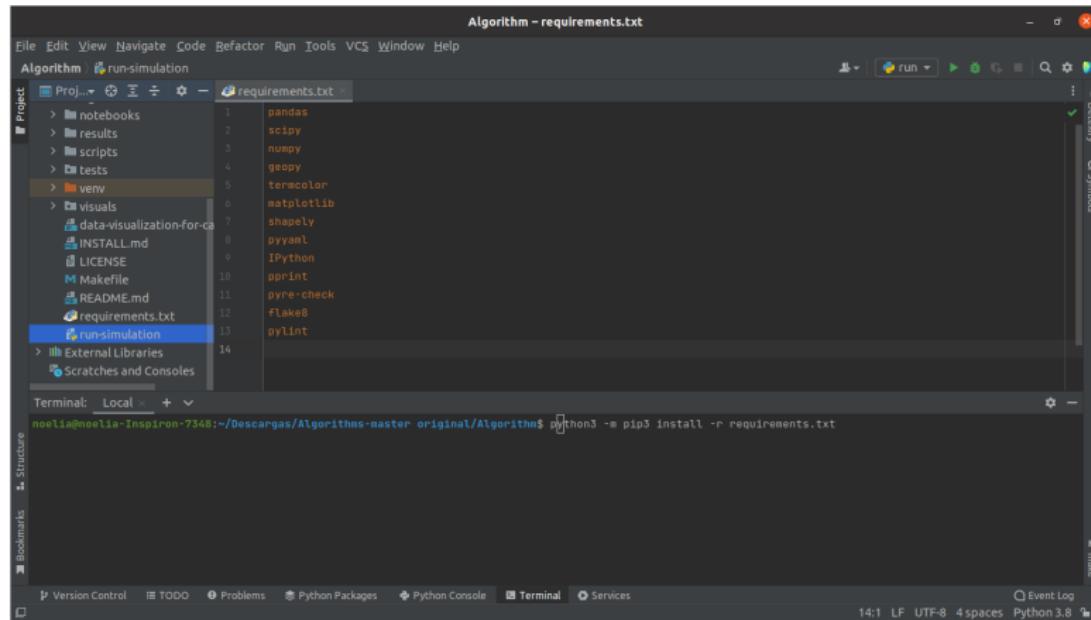


Figure 5: Installing Python packages

5. Simulator module (original version from UCSD)

A horizontal progress bar consisting of a thick dark grey segment followed by a thinner white segment.

Simulator module input

The simulator has an input where it requires the following information:

These are:

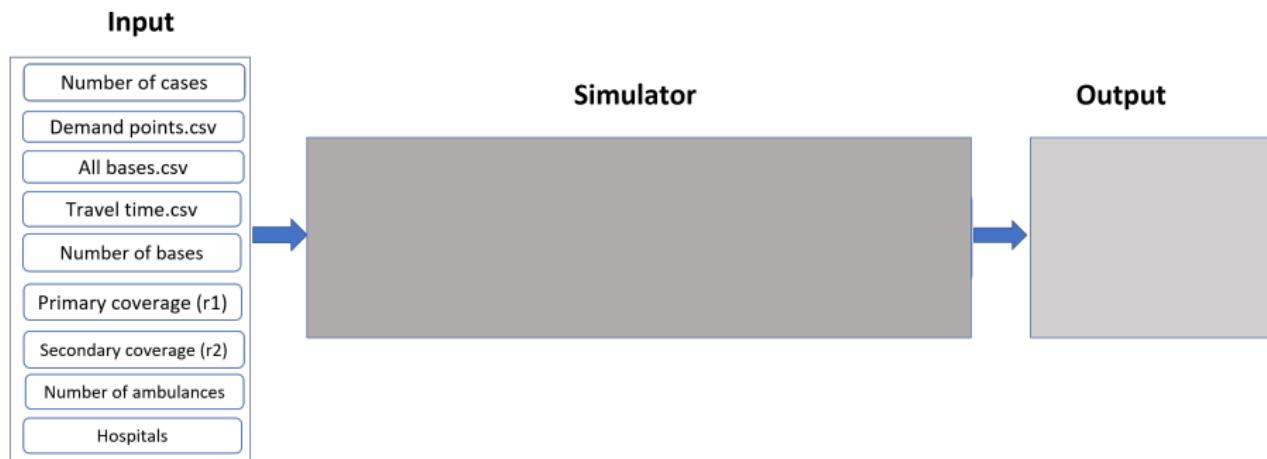


Figure 6: Simulator module.

Simulator module input

Number of cases : the number of cases (emergency calls) to be simulated.

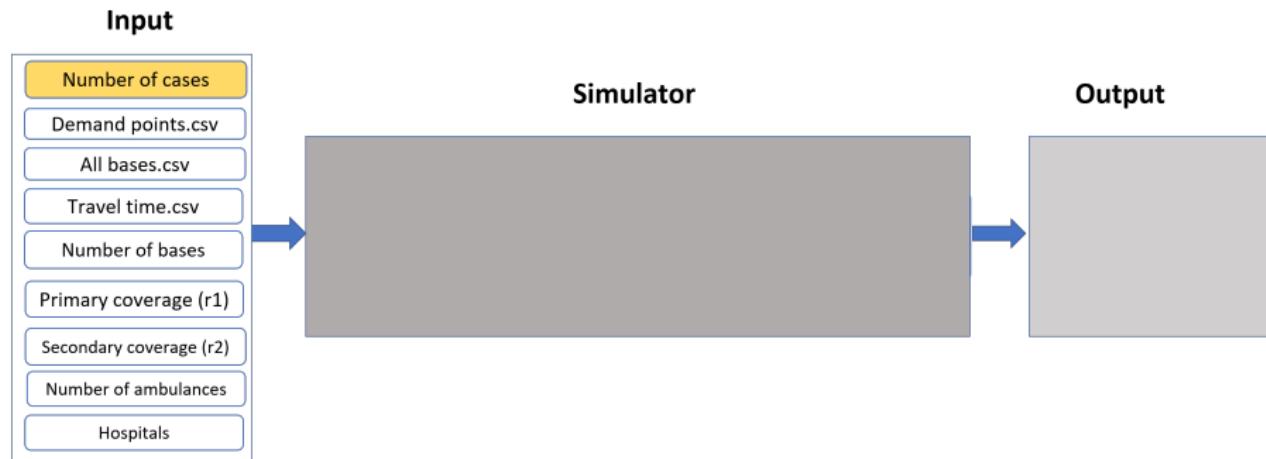


Figure 6: Simulator module.

Simulator module input

These files are available at:

<https://onx.la/7936c>

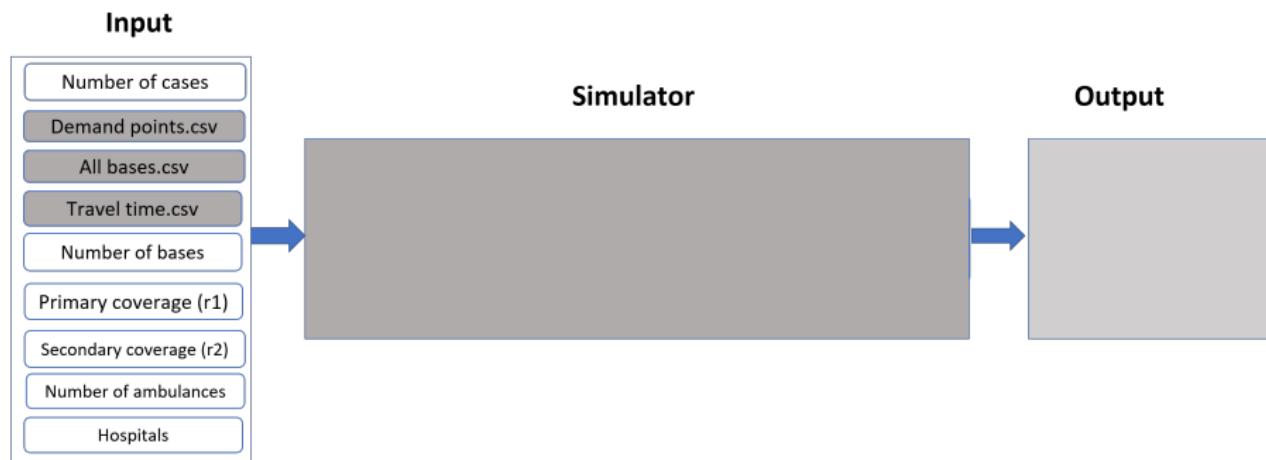


Figure 6: Simulator module.

Simulator module input

Demand points: are the group of calls determined by a clustering algorithm that generates the results in a *.csv file.

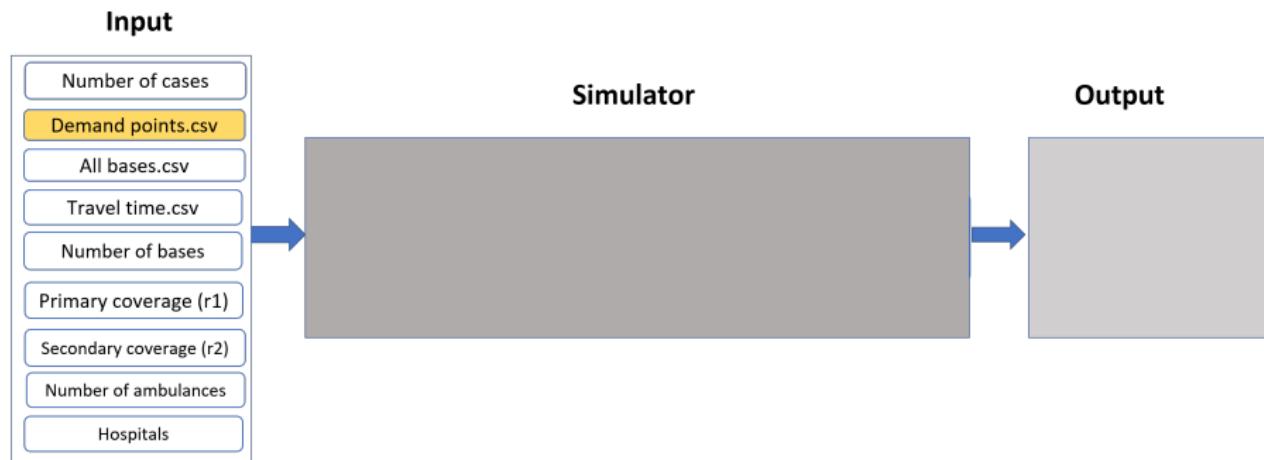


Figure 6: Simulator module.

Simulator module input

The file contains the geographic coordinate (latitude, longitude) of the location of the demand points.

The screenshot shows a LibreOffice Calc spreadsheet titled "demand_points.csv". The data consists of two columns: "latitude" and "longitude". The first few rows are as follows:

	A	B	C	D
1	latitude	longitude		
2	32.52463831	-117.0114142		
3	32.52625271	-117.0306702		
4	32.49288602	-116.8714818		
5	32.50885205	-116.8557693		
6	32.43844222	-117.0400494		
7	32.48782842	-117.0325632		
8	32.50295125	-116.926298		
9	32.50778	-116.895953		
10	32.42347711	-116.9597944		
11	32.39241625	-116.9403205		
12	32.54249471	-116.889966		
13	32.5061783	-117.0733549		
14	32.45912383	-116.876701		
15	32.47532105	-116.8287758		
16	32.53393858	-116.934083		
17	32.46824112	-116.9797883		
18	32.46935736	-116.9320901		
19	32.50743547	-116.9681504		
20				
21				
22				
23				
24				
25				
26				

Figure 7: Demand points file.

Simulator module input

All bases: 1691 locations (schools, fire stations, etc.) are considered candidate base locations and are available in a csv file.

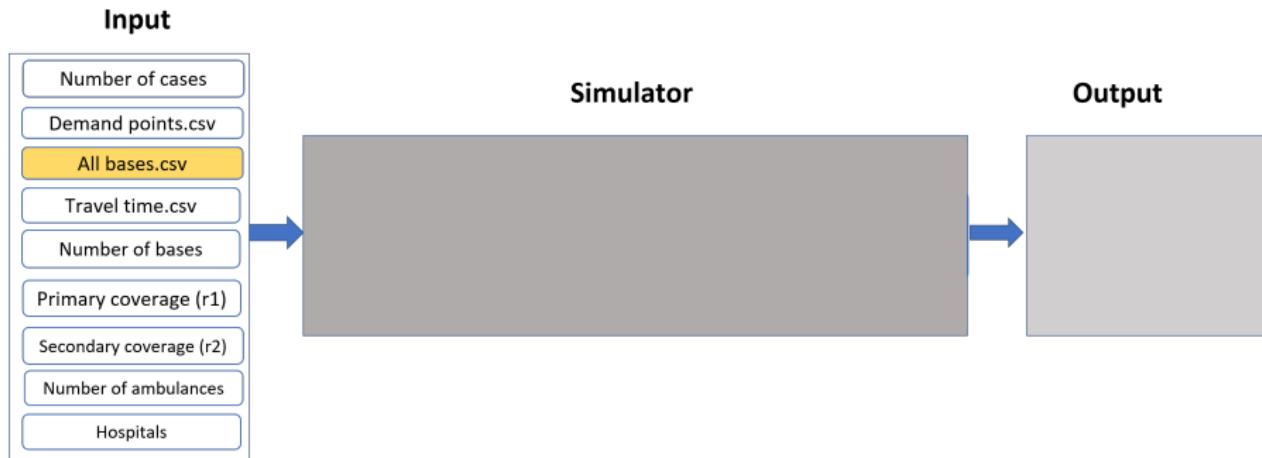
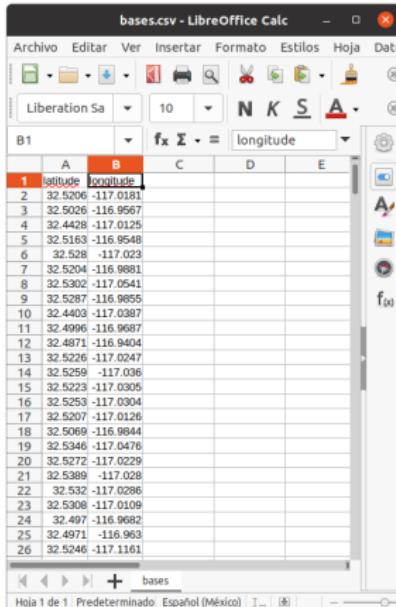


Figure 8: Simulator module.

Simulator module input

The file contains the geographic coordinate (latitude, longitude) of the location of the 1691 places, as possible bases.



The screenshot shows a LibreOffice Calc spreadsheet titled "bases.csv". The data is organized into two columns: "longitude" (Column A) and "latitude" (Column B). The first few rows of data are as follows:

	A	B
1	longitude	latitude
2	32.5206	-117.0181
3	32.5026	-116.9567
4	32.4428	-117.0125
5	32.5163	-116.9548
6	32.528	-117.023
7	32.5204	-116.9881
8	32.5302	-117.0541
9	32.5287	-116.9855
10	32.4403	-117.0387
11	32.4996	-116.9687
12	32.4871	-116.9404
13	32.5276	-117.0247
14	32.5289	-117.036
15	32.5223	-117.0305
16	32.5253	-117.0304
17	32.5207	-117.0126
18	32.5069	-116.9844
19	32.5346	-117.0476
20	32.5272	-117.0229
21	32.5389	-117.028
22	32.532	-117.0286
23	32.5308	-117.0109
24	32.487	-116.9682
25	32.4971	-116.963
26	32.5246	-117.1161

Figure 9: 1691 places as candidate bases.

Simulator module input

Travel time: A matrix of travel times (seconds) calculated with Google Maps, of the 1691 potential bases to each demand point, available in csv file.

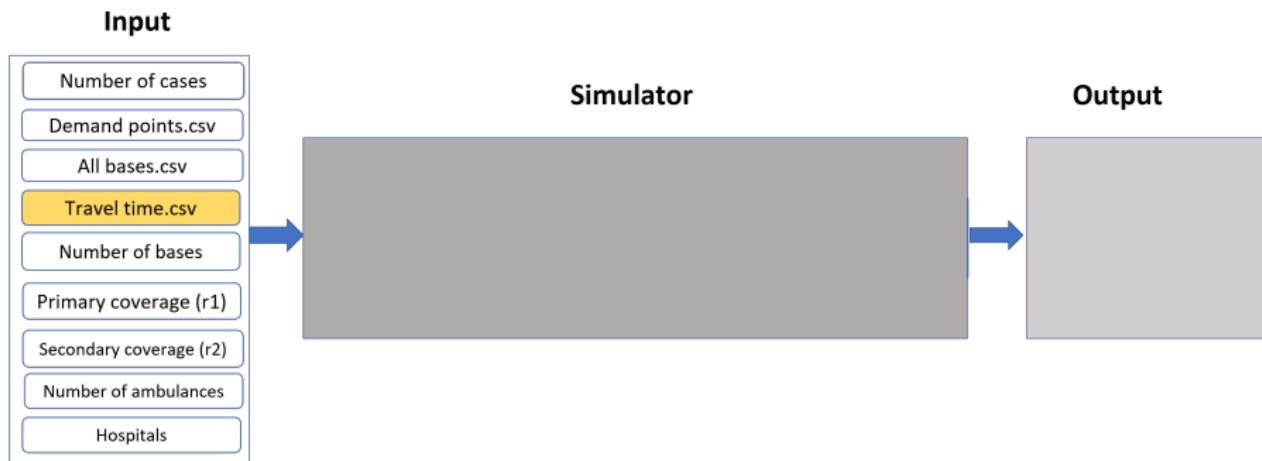
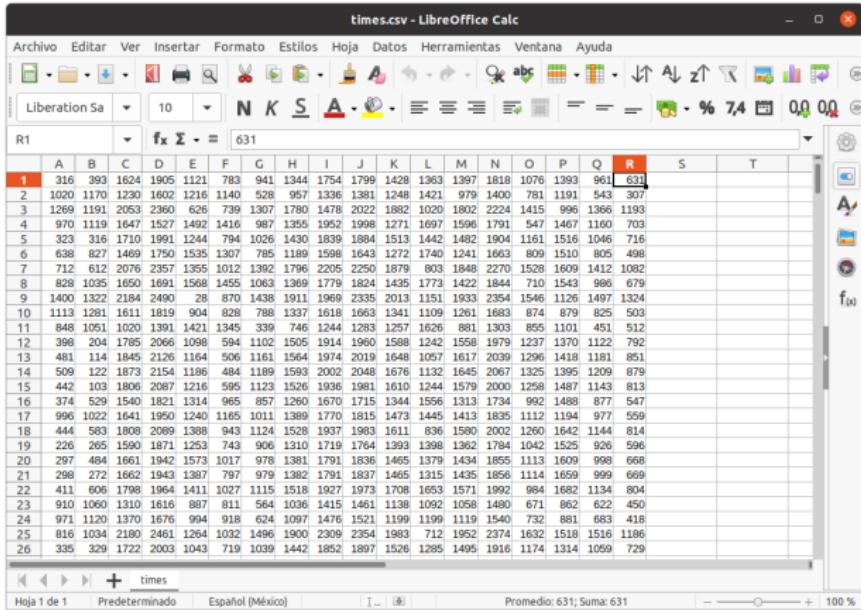


Figure 10: Simulator module.

Simulator module input

The file contains the travel times in seconds of the 1691 candidate bases to each demand point.



The screenshot shows a LibreOffice Calc spreadsheet titled "times.csv - LibreOffice Calc". The menu bar includes Archivo, Editar, Ver, Insertar, Formato, Estilos, Hoja, Datos, Herramientas, Ventana, and Ayuda. The toolbar includes various icons for file operations, cell selection, and data manipulation. The spreadsheet has a header row with columns labeled A through Q. The data starts at row 1, with the first few rows showing travel times between candidate bases (R1) and demand points (Q1). The file is currently set to 100% zoom.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	316	393	1624	1905	1121	783	941	1344	1754	1799	1428	1363	1397	1818	1076	1393	961	631		
2	1020	1230	1602	1216	1140	528	957	1336	1381	1248	1421	979	1400	781	1191	543	307			
3	1269	1191	2053	2360	626	739	1307	1780	1478	2022	1882	1020	1802	2224	1415	996	1366	1193		
4	970	1119	1647	1527	1492	1416	987	1355	1952	1998	1271	1697	1596	1791	547	1467	1160	703		
5	323	316	1710	1991	1244	794	1026	1430	1839	1884	1513	1442	1482	1904	1161	1516	1046	716		
6	638	827	1469	1750	1525	1307	785	1189	1598	1643	1272	1740	1241	1663	809	1510	805	498		
7	712	612	2076	2357	1355	1012	1392	1796	2205	2250	1879	803	1848	2270	1528	1609	1412	1082		
8	829	1035	1650	1691	1568	1455	1063	1369	1779	1824	1435	1773	1422	1844	710	1543	986	679		
9	1400	1322	2184	2490	28	870	1438	1911	1969	2335	2013	1151	1933	2354	1546	1126	1497	1324		
10	1113	1281	1611	1819	904	828	788	1337	1618	1663	1341	1109	1261	1683	874	879	825	503		
11	848	1051	1020	1391	1421	1345	339	746	1244	1283	1257	1626	881	1303	855	1101	451	512		
12	398	204	1765	2066	1098	594	1102	1505	1914	1960	1588	1242	1558	1979	1237	1370	1122	792		
13	481	114	1845	2126	1164	506	1161	1564	1974	2019	1648	1057	1617	2038	1296	1418	1181	851		
14	509	122	1873	2154	1186	484	1180	1593	2002	2048	1676	1132	1645	2067	1325	1395	1209	879		
15	442	103	1806	2087	1216	595	1123	1526	1938	1981	1610	1244	1579	2008	1258	1487	1143	813		
16	374	529	1540	1822	1314	965	857	1260	1670	1715	1344	1556	1313	1734	992	1488	877	547		
17	996	1022	1641	1950	1240	1165	1011	1389	1770	1815	1473	1445	1413	1838	1112	1194	977	559		
18	444	583	1808	2089	1388	943	1124	1528	1937	1983	1611	839	1580	2020	1260	1642	1144	814		
19	226	265	1590	1871	1253	743	904	1310	1716	1764	1393	1398	1362	1784	1042	1525	926	596		
20	297	484	1661	1942	1573	1017	978	1381	1791	1836	1465	1379	1434	1855	1113	1609	998	668		
21	298	272	1662	1943	1387	797	979	1382	1791	1837	1465	1315	1435	1856	1114	1659	999	669		
22	411	606	1798	1964	1411	1027	1115	1518	1927	1973	1708	1653	1571	1992	984	1682	1134	804		
23	910	1060	1370	1616	987	811	564	1036	1415	1461	1138	1092	1058	1480	671	862	622	450		
24	971	1120	1370	1676	994	918	624	1097	1476	1521	1199	1199	1119	1540	732	881	683	418		
25	816	1034	2180	2461	1284	1032	1496	1900	2309	2354	1983	712	1952	2374	1632	1518	1516	1186		
26	335	329	1722	2003	1043	719	1039	1442	1852	1897	1526	1285	1495	1916	1174	1314	1059	729		

Figure 11: Travel time file.

Simulator module input

Number of bases: It is the number of the best bases to be determined from the set of bases.

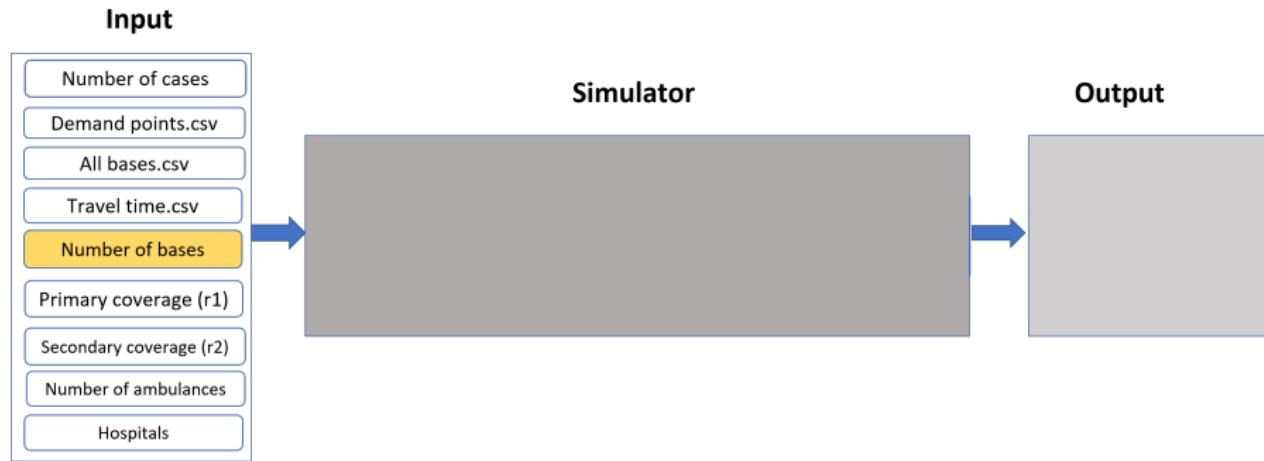


Figure 12: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

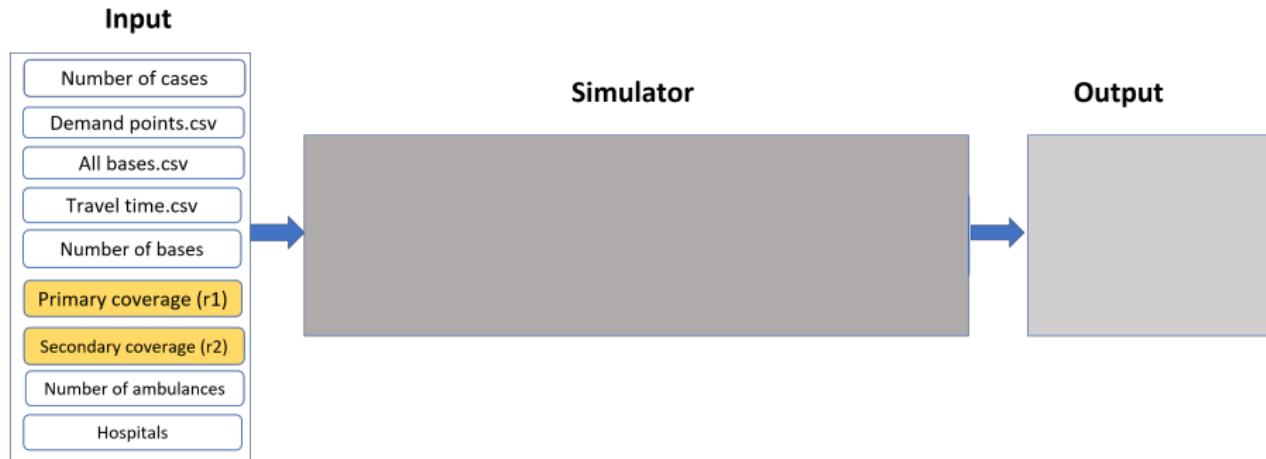


Figure 13: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

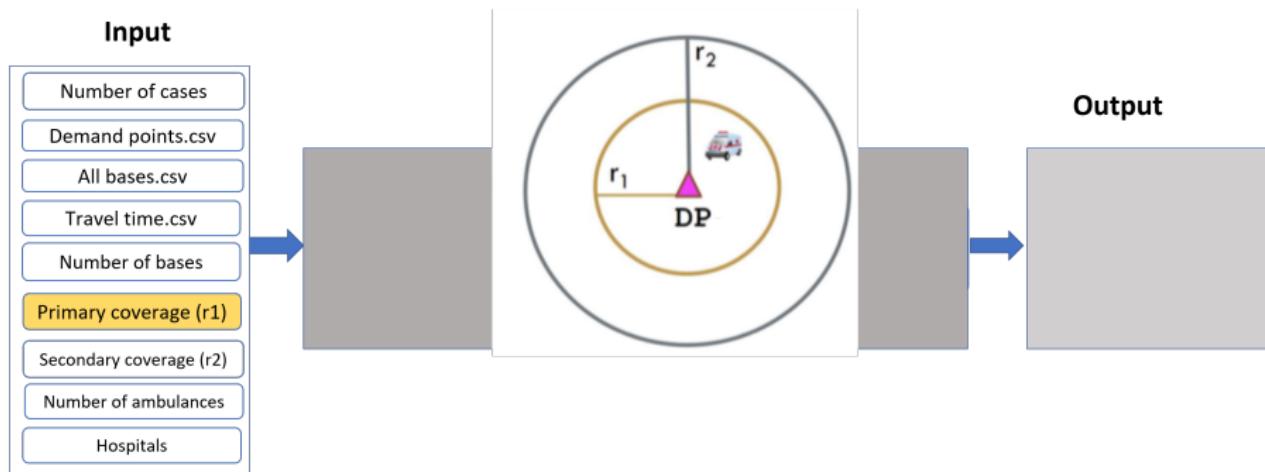


Figure 13: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

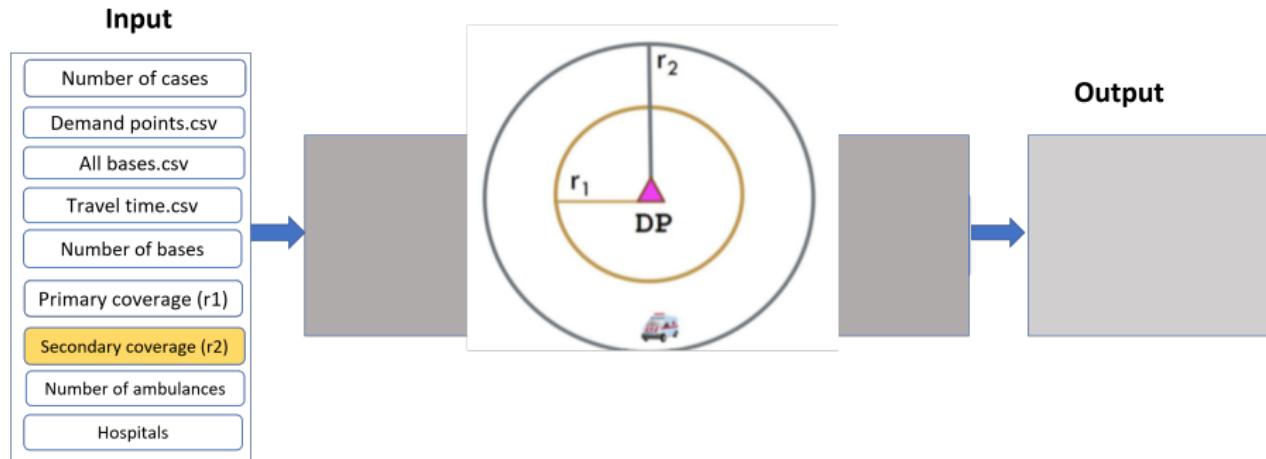


Figure 13: Simulator module.

Simulator module input

Number of ambulance: The number of ambulances available, this data must coincide with the number of bases established, so that each base has an ambulance.

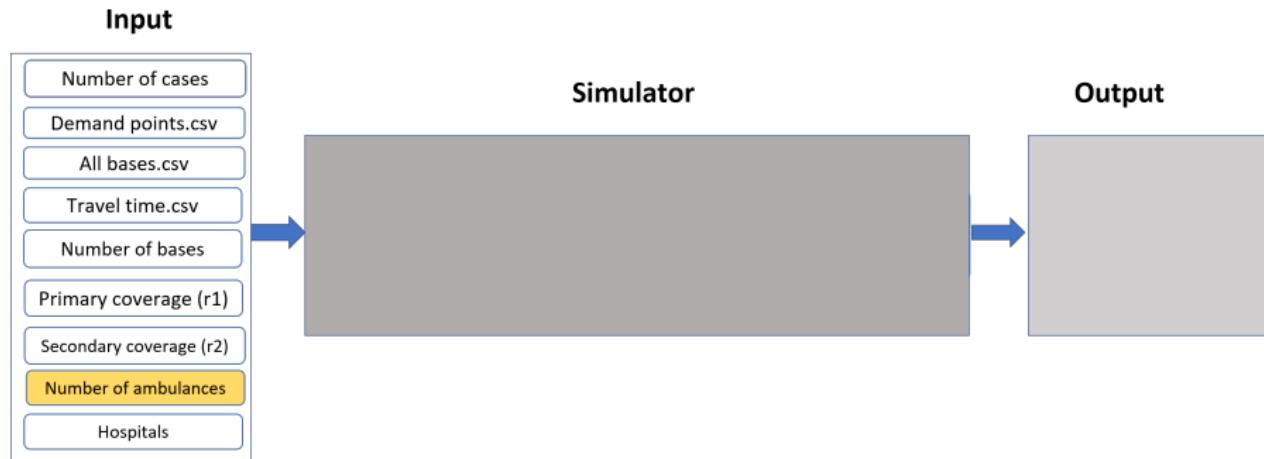


Figure 13: Simulator module.

Simulator module input

Hospitals: hospital locations (coordinates)

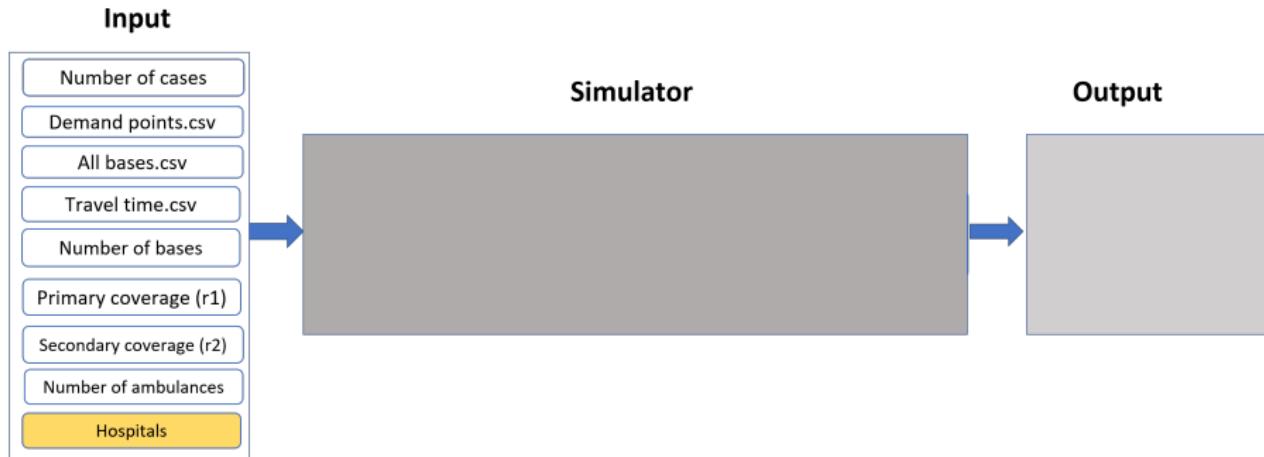


Figure 13: Simulator module.

Simulator input

The data required as input to the simulator is located in the **configurations –> example.yaml**

Watch video

Algorithm - example.yaml

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm configurations example.yaml
Project > example.yaml
  □ Project > □ + - example.yaml
    name: # File metadata, to be printed by the simulator
      Random Cases, Preset Bases, Preset Travel Times
    case_quantity: 50 # Generate 500 cases

    demands: # Read demands points in Tijeras area file
      class: ems.datasets.location.demand.demand.set.DemandSet
      filename: ../../data/cruz-roj-clean/demand_points.csv

    all_bases: # Read base points in Tijeras area file
      class: ems.datasets.location.base.base.set.BaseSet
      filename: ../../data/cruz-roj-clean/bases.csv

    # Travel times matrix between all bases and all demands
    class: ems.datasets.travel_times.TravelTimes
    filename: ../../data/cruz-roj-clean/times.csv
    origins: $all_bases
    destinations: $bases

    simulation_bases: # Selects a subset of bases for usage in the simulation (N-bases filtered)
      class: ems.datasets.location.base.filtered_base.set.FilteredBaseSet
      count: 12
      r1: 600
      r2: 840
    travel_time: 8tt

    ambulances: # Defines ambulances and assign bases in a round robin fashion from the simulation bases
      class: ems.datasets.ambulance.base_selected.AmbulanceSet
      count: 10
      selector:
        class: ems.algorithms.base_selectors.round_robin_selector.RoundRobinBaseSelector
        base_set: $simulation_bases

    hospitals: # Predefines 3 hospitals in the Tijeras area
      class: ems.datasets.location.hospital.hospital.set.HospitalSet
      latitudes: [32.520846012304, 32.5607, 32.518]
      longitudes: [-117.08979404798, -117.08371, -117.08708]

Document 1 / 1 | all_bases
Terminal: Local - + Run
Instrumentation: Instrumented, Baseline, Cases, Preset Cases, Preset Travel Times
Version Control: Q Find > Run > T1000 > Problems > Debug > Terminal > Python Packages > Python Console > Services
Low memory, the IDE is running low on memory and this might affect performance. Please consider increasing available heap. // Analyze memory use. Configure (yesterday 20:10) 10:31 LF UTF-8 2 spaces No JSON schema Python 3.8 (Algorithm)
```

Simulator process and output

In this block, the number of bases is determined from a set of bases. The number of **bases** must be equal to the **number of ambulances**, so that all bases are occupied. Finally, it is determined if there is primary coverage (r1) and secondary coverage (r2).

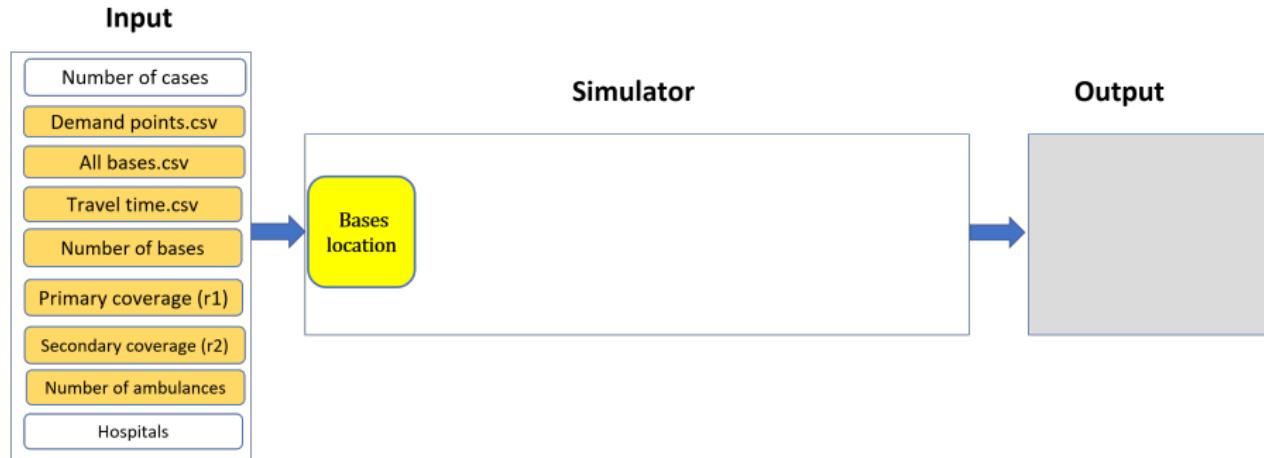


Figure 14: Simulator module.

Simulator process and output

With this block you will obtain the `chosen_ambulances.csv` file, which contains the coordinates of the selected bases.

The file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times - > chosen_bases.csv**.

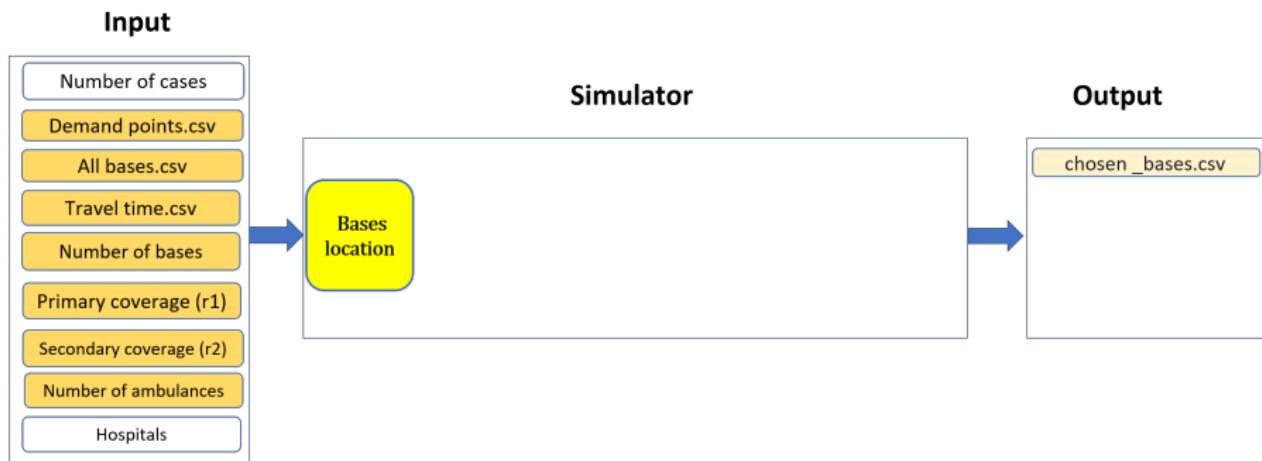
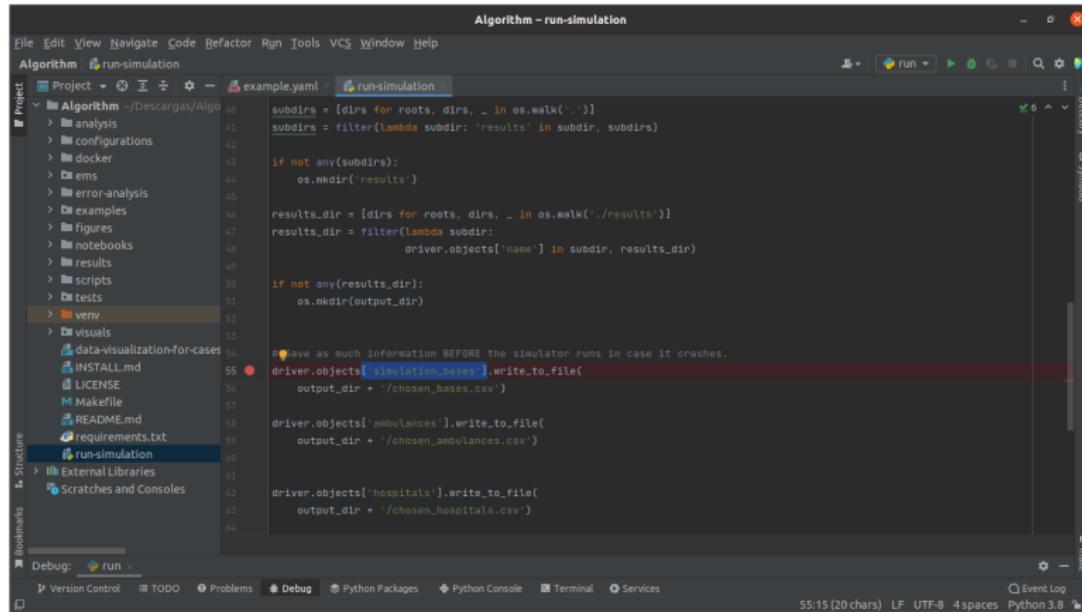


Figure 14: Simulator module.

Simulator process and output

The run-simulation.py file is responsible for calling the blocks in the example.yaml file following a sequence. In this case only the **simulation_bases** block.



The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Algorithm - run-simulation
- File Menu:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Standard, Run, Git, Diff, Details, Details
- Project Tree:** Shows the project structure with files like analysis, configurations, docker, ems, error-analysis, examples, figures, notebooks, results, scripts, tests, venv, and visuals.
- Code Editor:** Displays the run-simulation.py script. The code reads configuration from example.yaml and performs operations on subdirectories. It specifically handles the 'simulation_bases' block by writing to chosen_bases.csv and other CSV files for ambulances and hospitals.
- Bottom Status Bar:** Version Control, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, Event Log, 55:15 (20 chars), LF, UTF-8, 4 spaces, Python 3.8

```

Algorithm - run-simulation
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm - run-simulation
Project ~/Descargas/Algo 40
example.yaml x run-simulation
Algorithm
analysis 41
configurations 42
docker 43
ems 44
error-analysis 45
examples 46
figures 47
notebooks 48
results 49
scripts 50
tests 51
venv 52
visuals 53
data-visualization-for-cases 54
INSTALL.md 55
LICENSE 56
Makefile 57
README.md 58
requirements.txt 59
run-simulation 60
External Libraries 61
Scratches and Consoles 62
Structure 63
Bookmarks 64
Debug: run x
Version Control TODO Problems Debug Python Packages Python Console Terminal Services
Event Log 55:15 (20 chars) LF UTF-8 4 spaces Python 3.8

```

```

subdirs = [dirs for roots, dirs, _ in os.walk('.')]
subdirs = filter(lambda subdir: 'results' in subdir, subdirs)

if not any(subdirs):
    os.mkdir('results')

results_dir = [dirs for roots, dirs, _ in os.walk('./results')]
results_dir = filter(lambda subdir:
                     driver.objects['name'] in subdir, results_dir)

if not any(results_dir):
    os.mkdir(output_dir)

# Save as much information BEFORE the simulator runs in case it crashes.
driver.objects['simulation_bases'].write_to_file(
    output_dir + '/chosen_bases.csv')

driver.objects['ambulances'].write_to_file(
    output_dir + '/chosen_ambulances.csv')

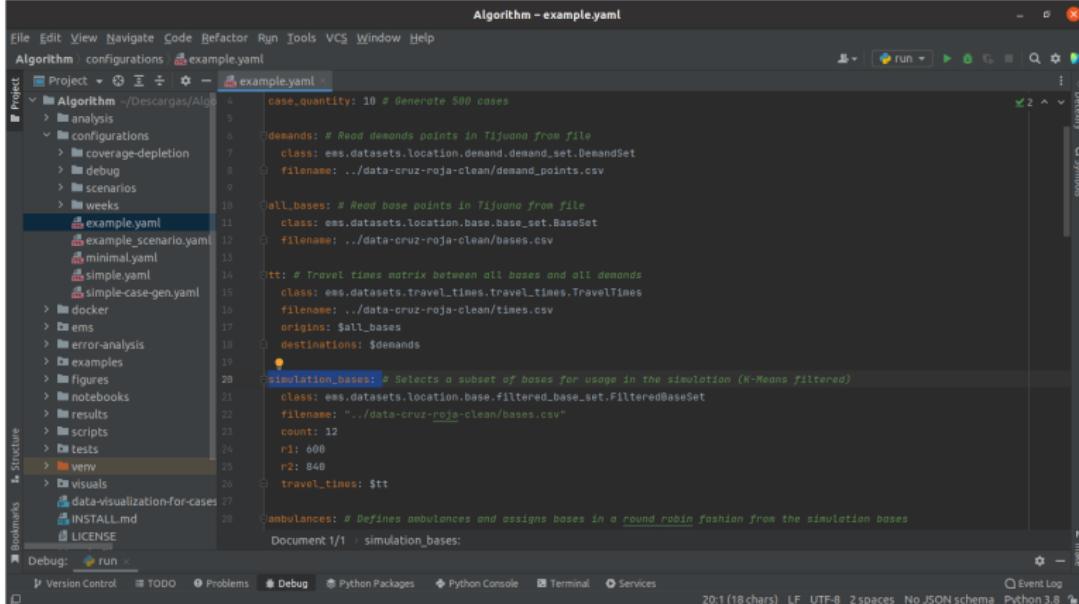
driver.objects['hospitals'].write_to_file(
    output_dir + '/chosen_hospitals.csv')

```

Figure 15: run-simulation.py source code

Simulator process and output

The run-simulation.py file is responsible for calling the blocks in the example.yaml file following a sequence. In this case only the **simulation_bases** block.



```

Algorithm - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm /-Descargas/Algo
Project - example.yaml
Algorithm
> analysis
> configurations
> coverage-depletion
> debug
> scenarios
> weeks
example.yaml
example_scenario.yaml
minimal.yaml
simple.yaml
simple-case-gen.yaml
docker
ems
error-analysis
examples
figures
notebooks
results
scripts
tests
venv
visuals
data-visualization-for-cases
INSTALL.md
LICENSE
Debug: run
Version Control TODO Debug Python Packages Python Console Terminal Services
20:1 (18 chars) LF UTF-8 2 spaces No JSON schema Python 3.8 Event Log
Document 1/1 > simulation_bases:

```

```

case_quantity: 10 # Generate 500 cases
demands: # Read demands points in Tijuana from file
class: ems.datasets.location.demand.demand_set.DemandSet
filename: ../data-cruz-roja-clean/demand_points.csv
all_bases: # Read base points in Tijuana from file
class: ems.datasets.location.base.base_set.BaseSet
filename: ../data-cruz-roja-clean/bases.csv
tt: # Travel times matrix between all bases and all demands
class: ems.datasets.travel_times.travel_times.TravelTimes
filename: ../data-cruz-roja-clean/times.csv
origins: $all_bases
destinations: $demands
simulation_bases: # Selects a subset of bases for usage in the simulation (K-Means filtered)
class: ems.datasets.location.base.filtered_base_set.FilteredBaseSet
filename: "../data-cruz-roja-clean/bases.csv"
count: 12
r1: 600
r2: 840
travel_times: $tt
ambulances: # Defines ambulances and assigns bases in a round robin fashion from the simulation bases

```

Figure 15: run-simulation.py source code

Simulator process and output

The video shows the files needed to run this first block, in this point, you can stop and get these first results.

Watch video

The screenshot shows a code editor interface with the following details:

- Project Tree:** Shows a project structure with files like `example.yaml`, `run-simulation.py`, `INSTALL.md`, `LICENSE`, `Makefile`, `README.md`, `requirements.txt`, and `venv`.
- Code Editor:** The `run-simulation.py` file is open, containing Python code. A red circular error marker is located at line 55, column 11, pointing to the opening brace of a block.
- Toolbar:** Includes standard icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and a Run dropdown.
- Bottom Bar:** Displays tabs for Debug, Run, Version Control, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, and a status bar showing the current time (55:15), file encoding (UTF-8), and workspace size (4 spaces).

Simulator process and output

With this block the ambulances are defined by assigning an ambulance to each base, saving the location coordinates of the ambulances in chosen_ambulances.csv, the file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times- > chosen_ambulances.csv**.

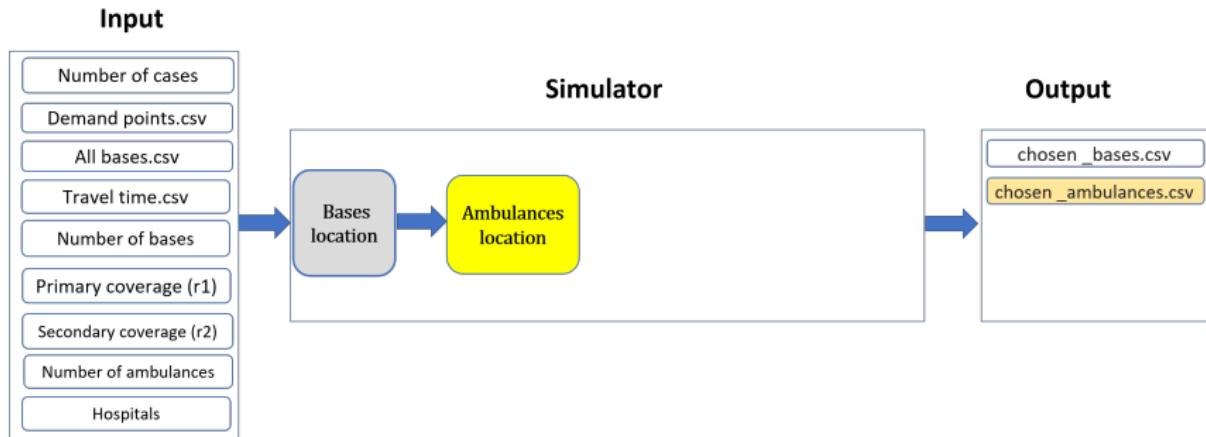
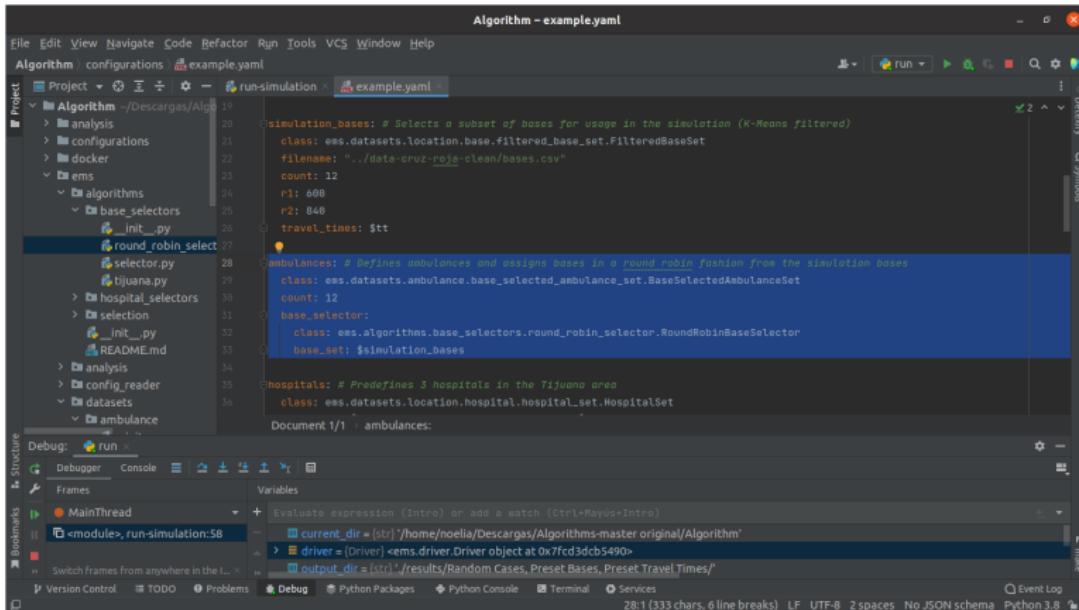


Figure 16: Simulator module.

Simulator process and output

In this case, with the information already generated from the base location block, the following **ambulances** block is executed.



The screenshot shows the PyCharm IDE interface with the file `Algorithm - example.yaml` open. The code defines several datasets and their configurations:

```

Algorithm - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm configurations example.yaml
Project run-simulation example.yaml
Algorithm /-/Descargas/Algo...
analysis
configurations
docker
ems
algorithms
base_selectors
_init_.py
round_robin_select
selector.py
tijuana.py
hospital_selectors
selection
_init_.py
README.md
analysis
config_reader
datasets
ambulance
simulation_bases: # Selects a subset of bases for usage in the simulation (K-Means filtered)
class: ems.datasets.location.base.filtered_base_set.FilteredBaseSet
filename: "../data/cruz-roja-clean/bases.csv"
count: 12
r1: 600
r2: 840
travel_times: $tt
ambulances: # Defines ambulances and assigns bases in a round robin fashion from the simulation bases
class: ems.datasets.ambulance.base_selected_ambulance_set.BaseSelectedAmbulanceSet
count: 12
base_selector:
class: ems.algorithms.base_selectors.round_robin_selector.RoundRobinBaseSelector
base_set: $simulation_bases
hospitals: # Predefines 3 hospitals in the Tijuana area
class: ems.datasets.location.hospital.hospital_set.HospitalSet
Document 1/1 ambulances:

```

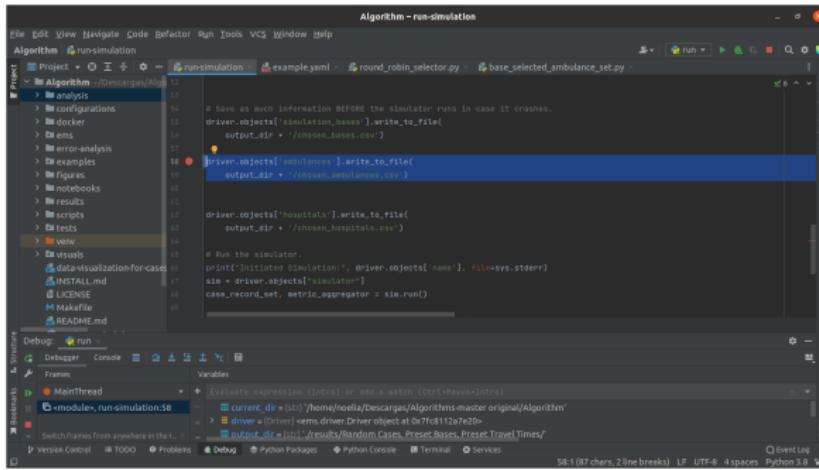
The code uses variables like `$tt`, `$base_set`, and `$simulation_bases`. The bottom of the editor shows an expression evaluation window with the current directory set to `/home/noelia/Descargas/Algorithms-master/original/Algorithm`.

Figure 17: example.yaml source code

Simulator process and output

The video shows the files needed to run this second block, in this point, you can stop and get the location of the ambulances which are the same as the location of the bases. Finally, when the file is saved, an additional field (capacity) is added, but this field will not be used in the experiments.

Watch video



```

Algorithm - run-simulation
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm /Desargues/Algorithm
Project run-simulation example.yaml round_robin_selector.py base_selected_ambulance_set.py
analysis
configurations
docker
ems
error-analysis
examples
figures
notebooks
results
scripts
tests
utils
data-visualization-for-cases
INSTALL.md
LICENSE
Makefile
README.md
Debug: run
MainThread
modules> run-simulation:58
Switch names from anywhere in this project...
Version Control TODO Problems Debug Python Packages Python Console Terminal Services
Overline
58:1 (87 chars, 2 line breaks), LF, UTF-8, 4 spaces, Python 3.8

# Save as much information BEFORE the simulator runs in case it crashes.
driver.objects['simulation_bases'].write_to_file(
    output_dir + '/chosen_bases.csv')

# Write the ambulances.
driver.objects['ambulances'].write_to_file(
    output_dir + '/chosen_ambulances.csv')

# Write the hospital states.
driver.objects['hospitalstates'].write_to_file(
    output_dir + '/chosen_hospitalstates.csv')

# Run the simulator.
print('Initiated simulation:', driver.objects['name'], file=sys.stderr)
sim = driver.objects['simulation']
case_record_set, metric_aggregator = sim.run()

```

Simulator process and output

In the hospital location block, the data are prepared to be the possible destinations of the patients to be transferred. At the same time it is managed in the `chosen_hospitals.csv` file.

The file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times - > chosen_hospitals.csv**.

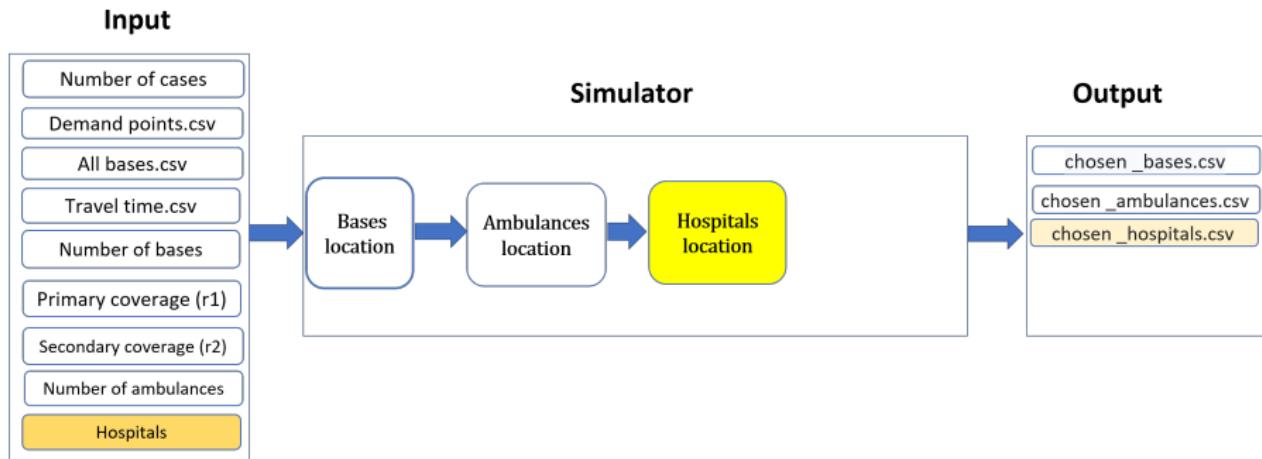
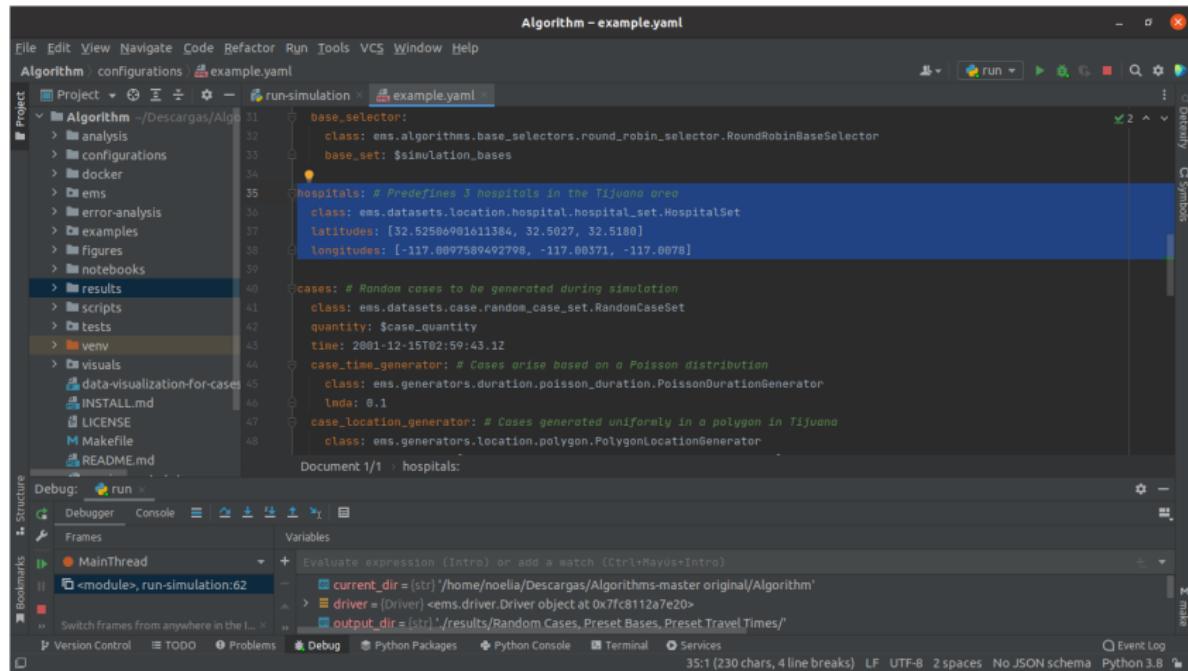


Figure 18: Simulator module.

Simulator process and output

This block indicates the coordinates of the hospitals.



```

Algorithm - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > configurations > example.yaml
Project Algorithm - /Descargas/Algo...
  - analysis
  - configurations
  - docker
  - ems
  - error-analysis
  - examples
  - Figures
  - notebooks
  - results
  - scripts
  - tests
  - venv
  - visuals
  - data-visualization-for-cases
  - INSTALL.md
  - LICENSE
  - Makefile
  - README.md
Algorithm > run-simulation > example.yaml
31   base_selector:
32     class: ems.algorithms.base_selectors.round_robin_selector.RoundRobinBaseSelector
33   base_set: $simulation_bases
34
35   hospitals: # Predefines 3 hospitals in the Tijuana area
36     class: ems.datasets.location.hospital.hospital_set.HospitalSet
37     latitudes: [32.52506901611384, 32.5027, 32.5180]
38     longitudes: [-117.0097589492798, -117.00371, -117.0078]
39
40   cases: # Random cases to be generated during simulation
41     class: ems.datasets.case.random_case_set.RandomCaseSet
42     quantity: $case_quantity
43     time: 2001-12-15T02:59:43.1Z
44     case_time_generator: # Cases arise based on a Poisson distribution
45       class: ems.generators.duration.poisson_duration.PoissonDurationGenerator
46       lmda: 0.1
47     case_location_generator: # Cases generated uniformly in a polygon in Tijuana
48       class: ems.generators.location.polygon.PolygonLocationGenerator
Document 1/1  hospitals:
Debug: run
Structure
Bookmarks
Frames
  MainThread
  <module>, run-simulation:62
  Switch frames From anywhere in the I...
Variables
  Evaluate expression (Intro) or add a watch (Ctrl+Mayús+Intro)
    current_dir = [str] '/home/noelia/Descargas/Algorithms-master/original/Algorithm'
    driver = [Driver] <ems.driver.Driver object at 0x7fc8112a7e20>
    output_dir = [str] '/results/Random Cases, Preset Bases, Preset Travel Times/'
Version Control
  TODO
  Problems
  Debug
  Python Packages
  Python Console
  Terminal
  Services
Event Log
  No JSON schema
  Python 3.8
  35:1 (230 chars, 4 line breaks) LF  UTF-8  2 spaces

```

Figure 19: example.yaml source code

Simulator process and output

The video shows the point that you can stop and see how hospital data is prepared for later use.

Watch video

```
# Save as much information BEFORE the simulator runs in case it crashes.
driver.objects['simulation_bases'].write_to_file(
    output_dir + '/chosen_bases.csv')

driver.objects['ambulances'].write_to_file(
    output_dir + '/chosen_ambulances.csv')

# Run the simulator.
print("Initiated Simulation:", driver.objects['name'], file=sys.stderr)
sim = driver.objects['simulator']
case.record_set, metric_aggregator = sim.run()
```

Simulator process and output

In this block several tasks are performed at the same time.

- Cases are generated.
- Ambulances are sent.
- Different events are generated.
- The percentage of coverage is calculated.

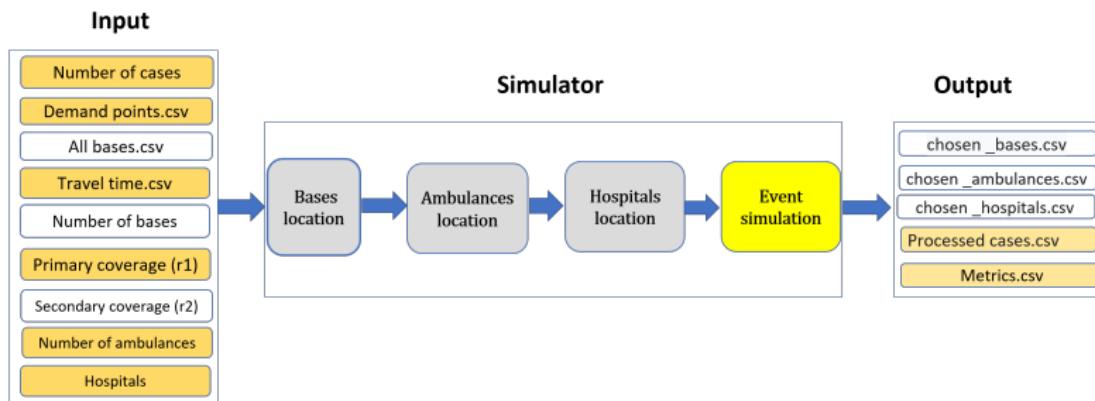
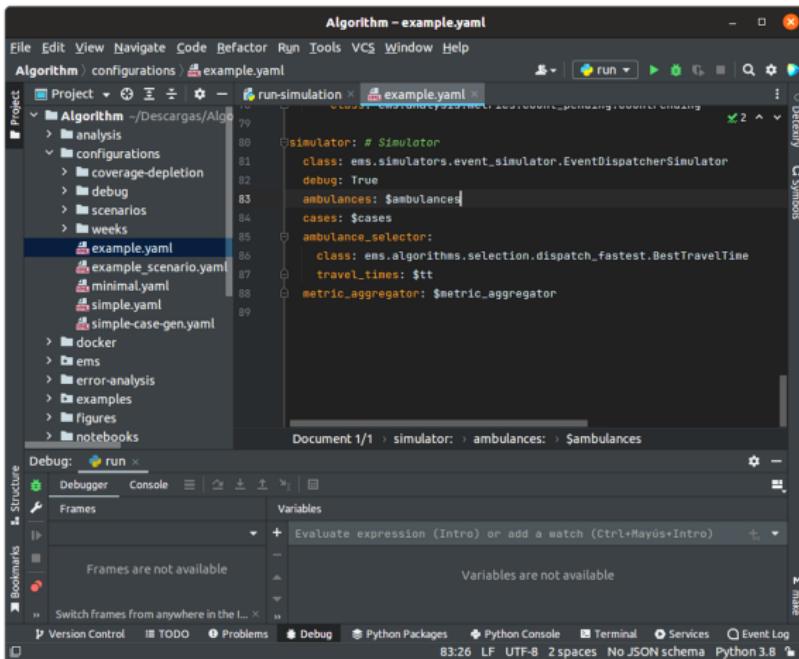


Figure 20: Simulator module.

Simulator process and output

This is the main block of the event simulation.



The screenshot shows a code editor window titled "Algorithm - example.yaml". The file content is as follows:

```
simulator: # Simulator
  class: ems.simulators.event_simulator.EventDispatcherSimulator
  debug: True
  ambulances: $ambulances
  cases: $cases
  ambulance_selector:
    class: ems.algorithms.selection.dispatch_fastest.BestTravelTime
    travel_times: $travel_times
  metric_aggregator: $metric_aggregator
```

The "Project" sidebar on the left lists several configuration files: analysis, configurations (coverage-depletion, debug, scenarios, weeks), docker, ems, error-analysis, examples, figures, and notebooks. The "example.yaml" file is selected in the project tree. The bottom status bar indicates "83:26 LF UTF-8 2 spaces No JSON schema Python 3.8".

Figure 21: example.yaml source code

Simulator process and output

The class of this block indicates the path to the file that performs all the processing.

The screenshot shows a code editor window titled "Algorithm - example.yaml". The left sidebar displays a project structure under the "Algorithm" folder, including subfolders like "analysis", "configurations", and "scenarios", along with files such as "example.yaml", "example_scenario.yaml", "minimal.yaml", "simple.yaml", and "simple-case-gen.yaml". The main editor area contains the YAML configuration file:

```
simulator: # Simulator
  class: ems.simulators.event_simulator.EventDispatcherSimulator
  debug: True
  ambulances: $ambulances
  cases: $cases
  ambulance_selector:
    class: ems.algorithms.selection.dispatch_fastest.BestTravelTime
    travel_times: $tt
  metric_aggregator: $metric_aggregator
```

The line "class: ems.simulators.event_simulator.EventDispatcherSimulator" is highlighted with a yellow box. Below the editor, the status bar shows "Document 1/1 > simulator: > ambulances". The bottom navigation bar includes tabs for "Version Control", "TODO", "Problems", "Debug", "Python Packages", "Python Console", "Terminal", "Services", "Event Log", and "Python 3.8".

Figure 21: example.yaml source code

Simulator process and output

Once inside the file, the first thing to do is to generate a case.

The screenshot shows a code editor interface with the following details:

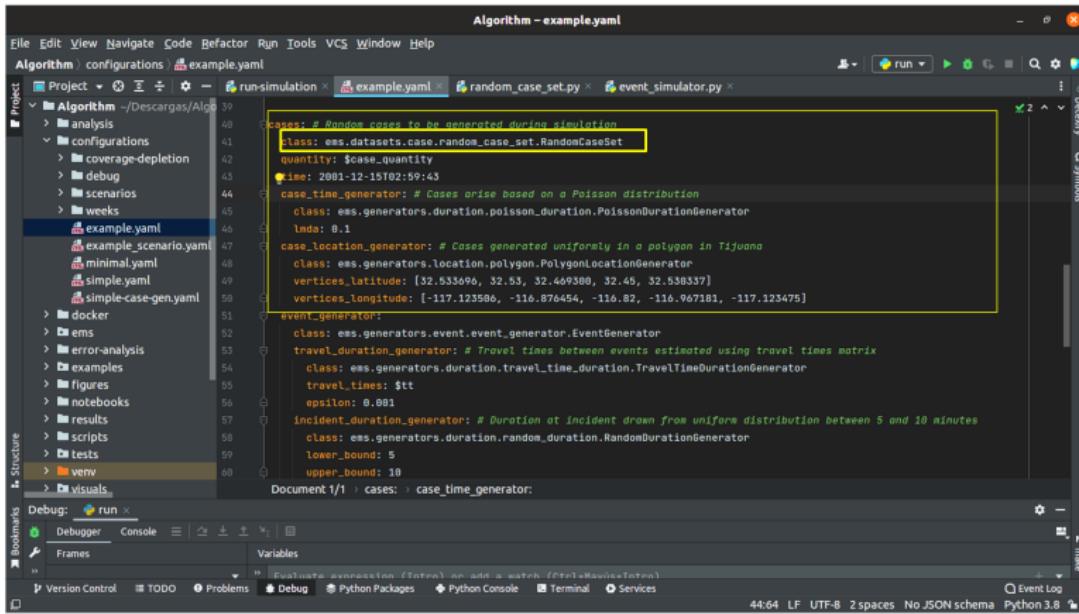
- Title Bar:** Algorithm - event_simulator.py
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Includes icons for run, debug, and other tools.
- Project Explorer:** Shows the project structure under "Algorithm":
 - ems
 - algorithms
 - analysis
 - config_reader
 - datasets
 - generators
 - models
 - scenarios
 - simulators
 - __init__.py
 - event_simulator.py
 - simulator.py
 - triggers
 - __init__.py
 - driver.py
 - utils.py
 - error-analysis
 - examples
 - Figures
 - notebooks
 - results
 - scripts
 - tests
 - venv
 - visuals
- Code Editor:** Displays the event_simulator.py file content. A specific line of code is highlighted:

```
        next_case = next(case_iterator)
```
- Status Bar:** Shows the current file is EventDispatcherSimulator > run()
- Bottom Navigation:** Debug: run, Version Control, TODO, Problems, Python Packages, Python Console, Terminal, Services, Event Log, 61:7 (33 chars), LF, UTF-8, 4 spaces, Python 3.8

Figure 22: event_simulator.py source code

Simulator process and output

This is the block in charge of generating this new case, the class indicates the file path where this process is carried out.



```

Algorithm - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > configurations > example.yaml
Project: Algorithm - /Descargas/Algorithm
  > analysis
  > configurations
    > coverage-depletion
    > debug
    > scenarios
    > weeks
      > example.yaml
        > example_scenario.yaml
        > minimal.yaml
        > simple.yaml
        > simple-case-gen.yaml
      > docker
      > ems
      > error-analysis
      > examples
      > figures
      > notebooks
      > results
      > scripts
      > tests
      > venv
      > visuals
Structure > example.yaml
  cases: # Random cases to be generated during simulation
    class: ems.datasets.case.random_case_set.RandomCaseSet
    quantity: $case.quantity
    time: 2001-12-19T02:59:43
    case_time_generator: # Cases arise based on a Poisson distribution
      class: ems.generators.duration.poisson_duration.PoissonDurationGenerator
      lmda: 0.1
    case.location_generator: # Cases generated uniformly in a polygon in Tijuana
      class: ems.generators.location.polygon.PolygonLocationGenerator
      vertices_latitude: [32.553696, 32.65, 32.469386, 32.45, 32.550337]
      vertices_longitude: [-117.123586, -116.876454, -116.82, -116.967181, -117.123475]
    event_generator:
      class: ems.generators.event.event_generator.EventGenerator
      travel_duration_generator: # Travel times between events estimated using travel times matrix
        class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
        travel_times: $tt
        epsilon: 0.001
      incident_duration_generator: # Duration of incident drawn from uniform distribution between 5 and 10 minutes
        class: ems.generators.duration.random_duration.RandomDurationGenerator
        lower_bound: 5
        upper_bound: 10
Document 1/1 > cases: > case_time_generator:
Debug: run >
  Debugger Console Variables
  Frames
  Evaluate expression (IntelliJ) or add a watch (Ctrl+Shift+Totem)
Version Control TODO Problems Debug Python Packages Python Console Terminal Services
  Event Log
  44:64 LF UTF-8 2 spaces No JSON schema Python 3.8

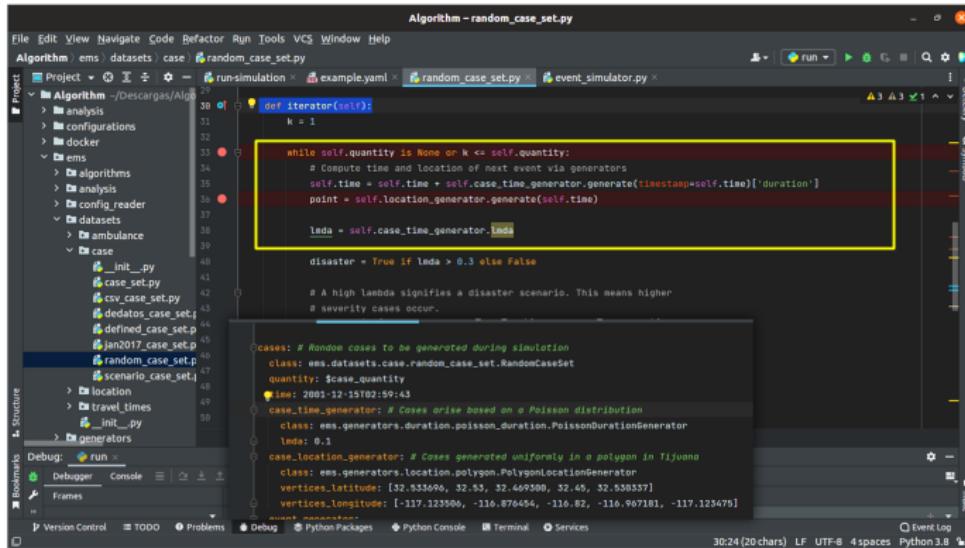
```

Figure 23: example.yaml source code

Simulator process and output

Within this file, a new case is generated with:

- A random time for the call.
- A random location of the call within a previously set area.



```

Algorithm - random_case_set.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm ems datasets case random_case_set.py
Project analysis configurations docker
ems algorithms analysis config_reader datasets ambulance case
__init__.py case_set.py csv_case_set.py dedatos_case_set.py defined_case_set.py jan2017_case_set.py random_case_set.py scenario_case_set.py
location travel_times __init__.py generators
def iterator(self):
    k = 1
    while self.quantity is None or k <= self.quantity:
        # Compute time and location of next event via generators
        self.time = self.time + self.case_time_generator.generate(timestamp=self.time)['duration']
        point = self.location_generator.generate(self.time)

        lmda = self.case_time_generator.lmda
        disaster = True if lmda > 0.3 else False

        # A high lmda signifies a disaster scenario. This means higher
        # severity cases occur.

cases: # Random cases to be generated during simulation
class: ems.datasets.case.random_case_set.RandomCaseSet
quantity: $case_quantity
time: 2003-12-15T02:59:43
case_time_generator: # Cases arise based on a Poisson distribution
class: ems.generators.duration.poisson_duration.PoissonDurationGenerator
lmda: 0.1
case_location_generator: # Cases generated uniformly in a polygon in Tijuana
class: ems.generators.location.polygon.PolygonLocationGenerator
vertices_latitude: [32.553696, 32.55, 32.449200, 32.45, 32.558537]
vertices_longitude: [-117.123500, -116.876454, -116.82, -116.997181, -117.123547]

```

Figure 24: random_case_set.py source code.

Simulator process and output

The events that the case may have are generated.

```
Algorithm - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > configurations > example.yaml
Proj... run-simulation event_simulator.py event_generator.py example.yaml
Algorithm ~/Descargas/A
> analysis
configurations
> coverage-depletion
> debug
> scenarios
> weeks
example.yaml
example_scenario.yaml
minimallyaml
simple.yaml
simple-case-gen.yaml
> docker
> ems
> error-analysis
> examples
> figures
> notebooks
> results
> scripts
> tests
> venv
visuals
data-visualization-for-ca
INSTALL.md
LICENSE
Document 1/1 > cases: > event_generator: > travel_duration_generator: > class: > ems.generators.durat...
Debug: run
Version Control Find TODO Problems Debug Python Packages Python Console Terminal Services
54:86 LF UTF-8 2 spaces No JSON schema Python 3.8
Structure Bookmarks
Algorithm example.yaml
cases: # Random cases to be generated during simulation
  class: ems.datasets.case.random_case_set.RandomCaseSet
  quantity: $case_quantity
  time: 2001-12-15T02:59:43
  case_time_generator: # Cases arise based on a Poisson distribution
    class: ems.generators.duration.poisson_duration.PoissonDurationGenerator
    lmda: 0.1
  case_location_generator: # Cases generated uniformly in a polygon in Tijuana
    class: ems.generators.location.polygon.PolygonLocationGenerator
    vertices_latitude: [32.533696, 32.53, 32.469300, 32.45, 32.530337]
    vertices_longitude: [-117.123506, -116.876454, -116.82, -116.967181, -117.123475]
  event_generator:
    class: ems.generators.event.event_generator.EventGenerator
    travel_duration_generator: # Travel times between events estimated using travel times matrix
      class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
      travel_times: $tt
      epsilon: 0.001
    incident_duration_generator: # Duration at incident drawn from uniform distribution between 5 and 10 minutes
      class: ems.generators.duration.random_duration.RandomDurationGenerator
      lower_bound: 5
      upper_bound: 10
    hospital_duration_generator: # Duration at hospital drawn from uniform distribution between 5 and 10 minutes
      class: ems.generators.duration.random_duration.RandomDurationGenerator
      lower_bound: 5
      upper_bound: 10
```

Figure 25: example.yaml source code.

Simulator process and output

Events are defined in this file.

The screenshot shows a Python code editor interface with the following details:

- Title Bar:** Algorithm - event_generator.py
- Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Toolbar:** Includes icons for run, search, and file operations.
- Project Explorer:** Shows the project structure:
 - Algorithm
 - analysis
 - configurations
 - docker
 - ems
 - algorithms
 - analysis
 - config_reader
 - datasets
 - generators
 - duration
 - event
 - event_generator
 - location
 - init_.py
 - models
 - scenarios
 - simulators
 - triggers
 - init_.py
 - driver.py
 - utils.py
 - error-analysis
 - examples
- Code Editor:** Displays the event_generator.py file content:

```
destination = None
duration = 0

if event_type == EventType.TO INCIDENT:
    destination = incident_location
    duration = self.travel_duration_generator.generate(ambulance=ambulance,
                                                       destination=incident_location,
                                                       timestamp=timestamp)

elif event_type == EventType.AT INCIDENT:
    destination = incident_location
    duration = self.incident_duration_generator.generate(ambulance=ambulance,
                                                          destination=incident.location,
                                                          timestamp=timestamp)

elif event_type == EventType.TO BASE:
    destination = ambulance.base
    duration = self.travel_duration_generator.generate(ambulance=ambulance,
                                                       destination=destination,
                                                       timestamp=timestamp)

elif event_type == EventType.TO HOSPITAL or event_type == EventType.AT HOSPITAL:

    if not hospital_location:
        destination = self.hospital_selector.select(timestamp=timestamp,
                                                     ambulance=ambulance)

    else:
```
- Bottom Status Bar:** Debug: run, Version Control, Find, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, 29:1 LF UFT-8 4 spaces Python 3.8, Event Log.

Figure 26: event_generator.py source code.

Simulator process and output

A case can have different events and depending on this, the status is established.

Cases	Event	Status
1	Arrive to incident	Ongoing
2	Attending to incident	Ongoing
3	Heading to hospital	Ongoing
4	Leave patient	Ongoing
5	Returning to base	Done

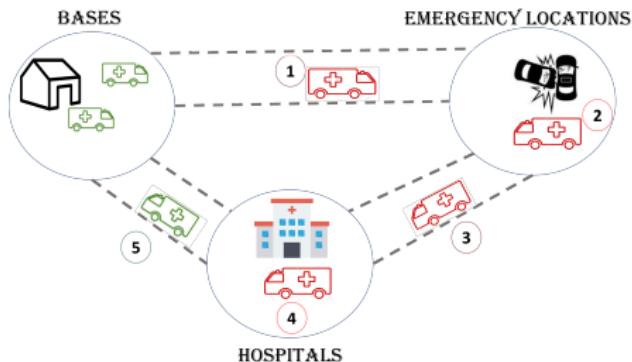
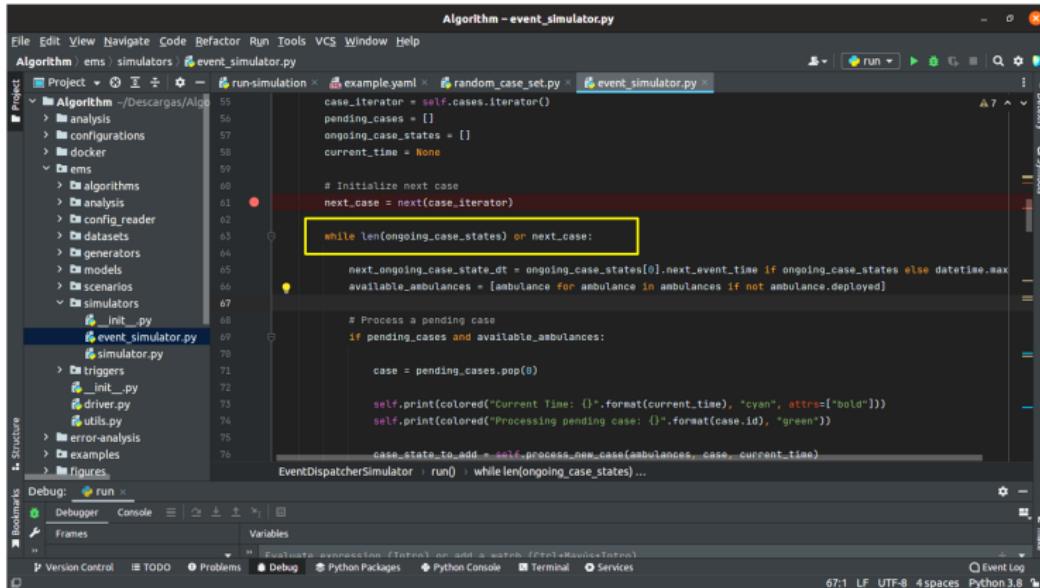


Figure 27: Case events.

Simulator process and output

Continuing with the main file, the cases start running as long as the first condition is met:

if there are cases in ongoing or a new case has been generated.



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Algorithm - /Descargas/Algorithm
- File:** event_simulator.py
- Code Content:**

```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm / eems / simulators / event_simulator.py
Project run-simulation example.yaml random_case_set.py event_simulator.py
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
case_iterator = self.cases.iterator()
pending_cases = []
ongoing_case_states = []
current_time = None

# Initialize next case
next_case = next(case_iterator)

while len(ongoing_case_states) or next_case:
    next_ongoing_case_state_dt = ongoing_case_states[0].next_event_time if ongoing_case_states else datetime.max
    available_ambulances = [ambulance for ambulance in ambulances if not ambulance.deployed]

    # Process a pending case
    if pending_cases and available_ambulances:
        case = pending_cases.pop(0)

        self.print(colored("Current Time: {}.".format(current_time), "cyan", attrs=["bold"]))
        self.print(colored("Processing pending case: {}.".format(case.id), "green"))

        case.state_to_add = self._process_new_case(ambulances, case, current_time)
EventDispatcherSimulator > run() > while len(ongoing_case_states) ...

```
- Toolbars and Panels:** Project, Run, Structure, Bookmarks, Debug, Version Control, TODO, Problems, Python Packages, Python Console, Terminal, Services.
- Status Bar:** 67:1 LF UTF-8 4 spaces Python 3.8

Figure 28: event_simulator.py source code..

Simulator process and output

The available ambulances are verified.

The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Project Tree:** Algorithm, analysis, configurations, docker, ems, datasets, generators, models, scenarios, simulators, triggers, error-analysis, examples, figures.
- Current File:** event_simulator.py
- Code Editor:** The code handles case iteration and processing. A specific section of the code is highlighted with a yellow box:

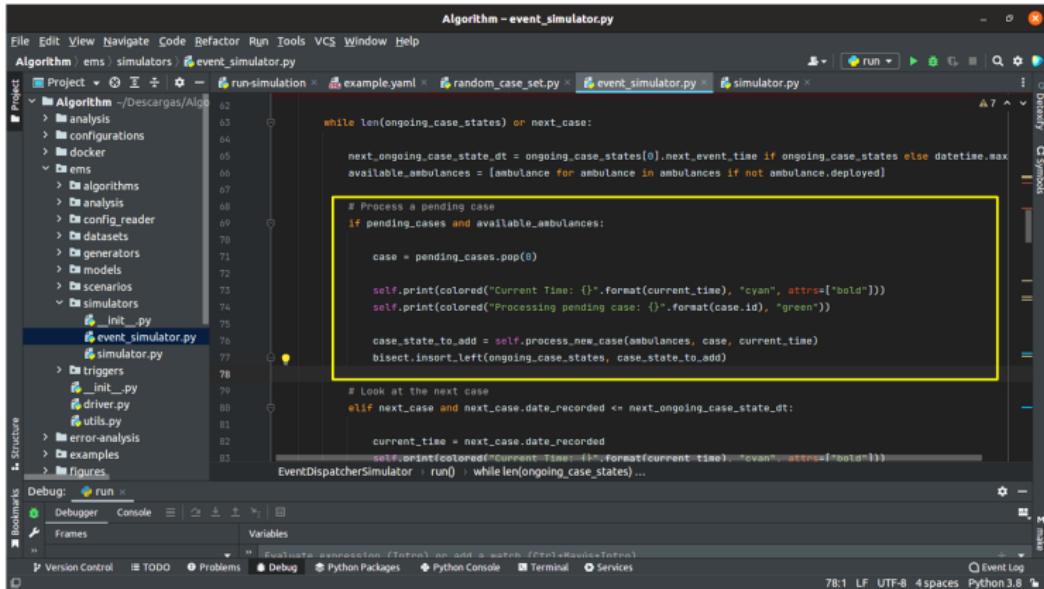
```
next_ongoing_case_state_dt = ongoing_case_states[0].next_event_time if ongoing_case_states else datetime.datetime.max
available_ambulances = [ambulance for ambulance in ambulances if not ambulance.deployed]
```
- Run Tab:** Shows a run configuration for 'run-simulation'.
- Debug Tab:** Shows a 'run' configuration with a 'Debugger' tab selected.
- Bottom Bar:** Version Control, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, Event Log.

Figure 28: event_simulator.py source code..

Simulator process and output

Moving on to the next condition:

If there are pending cases to attend and ambulances available.



The screenshot shows the PyCharm IDE interface with the file `Algorithm - event_simulator.py` open. The code is part of a larger project structure under the `Algorithm` package. A yellow box highlights the following section of code:

```

while len(ongoing_case_states) or next_case:
    next_ongoing_case_state_dt = ongoing_case_states[0].next_event_time if ongoing_case_states else datetime.datetime.max
    available_ambulances = [ambulance for ambulance in ambulances if not ambulance.deployed]

    # Process a pending case
    if pending_cases and available_ambulances:
        case = pending_cases.pop(0)

        self.print(colored("Current Time: {}".format(current_time), "cyan", attrs=[bold]))
        self.print(colored("Processing pending case: {}".format(case.id), "green"))

        case.state_to_add = self._process_new_case(ambulances, case, current_time)
        bisect.insort_left(ongoing_case_states, case.state_to_add)

        # Look at the next case
        elif next_case and next_case.date_recorded <= next_ongoing_case_state_dt:
            current_time = next_case.date_recorded
            self.print(colored("Current Time: {}".format(current_time), "cyan", attrs=[bold]))

```

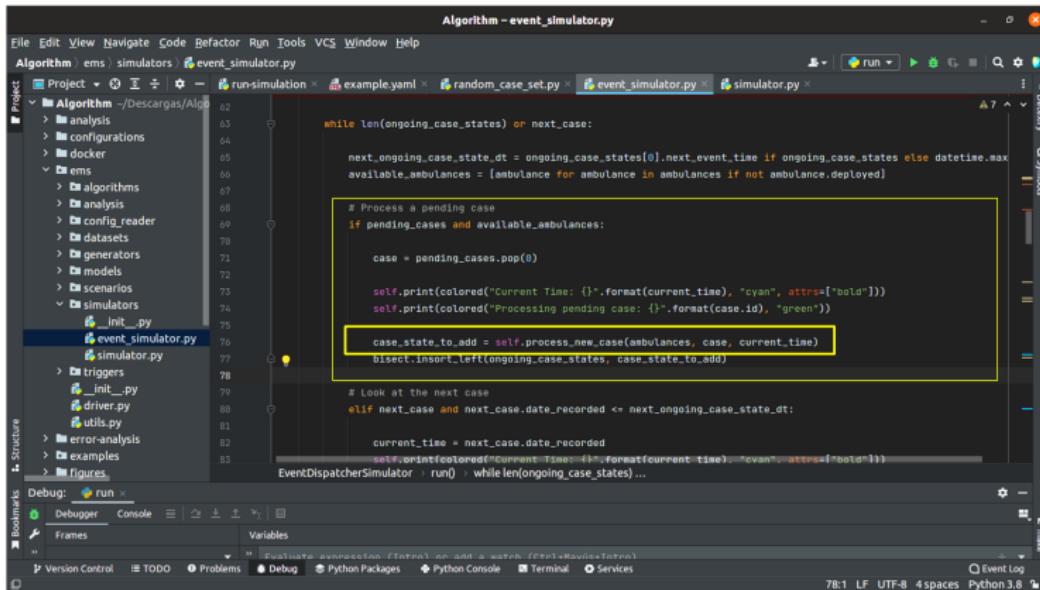
The code implements a simulation loop. It processes pending cases if there are any pending cases and available ambulances. For each pending case, it prints the current time and the ID of the case being processed. It then updates the case's state and inserts it into the list of ongoing cases. Finally, it checks for the next case and updates the current time if the next case's recorded date is earlier than or equal to the current ongoing case's state date.

Figure 28: `event_simulator.py` source code..

Simulator process and output

Moving on to the next:

Start a new case.



```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm.ems simulators event_simulator.py
Project /-Descargas/Algorithm
  - analysis
  - configurations
  - docker
  - ems
    - algorithms
    - analysis
    - config_reader
    - datasets
    - generators
    - models
    - scenarios
    - simulators
      - __init__.py
      - event_simulator.py
      - simulator.py
    - triggers
    - __init__.py
    - driver.py
    - utils.py
  - error-analysis
  - examples
  - figures
  - Structure
  - Bookmarks
Debug: run
  Debugger Console
  Frames Variables
  Evaluate expression (Threads) An add a watch (Threads) Modules (Threads)
Version Control TODO Problems Debug Python Packages Python Console Terminal Services Event Log
78:1 LF UTF-8 4spaces Python 3.8
while len(ongoing_case_states) or next_case:
    next_ongoing_case_state_dt = ongoing_case_states[0].next_event_time if ongoing_case_states else datetime.datetime.max
    available_ambulances = [ambulance for ambulance in ambulances if not ambulance.deployed]

    # Process a pending case
    if pending_cases and available_ambulances:
        case = pending_cases.pop(0)

        self.print(colored("Current Time: {}".format(current_time), "cyan", attrs=[bold]))
        self.print(colored("Processing pending case: {}".format(case.id), "green"))

        case.state_to_add = self.process_new_case(ambulances, case, current_time)
        bisect.insort_left(ongoing_case_states, case.state_to_add)

    # Look at the next case
    elif next_case and next_case.date_recorded <= next_ongoing_case_state_dt:
        current_time = next_case.date_recorded
        self.print(colored("Current Time: {} >= {} (current time)".format(current_time), "cyan", attrs=[bold]))
        self.print("EventDispatcherSimulator > run() > while(len(ongoing_case_states) ..."))

```

Figure 28: event_simulator.py source code..

Simulator process and output

When a new case is started, this section of the code is executed.

```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > ems > simulators > event_simulator.py
Project > Algorithm /Descargas/Algo 140
> analysis 141
> configurations 142
> docker 143
> ems 144
> algorithms 145
> analysis 146
> config_reader 147
> datasets 148
> generators 149
> models 150
> scenarios 151
> simulators 152
    > __init__.py 153
    > event_simulator.py 154
    > simulator.py 155
> triggers 156
    > __init__.py 157
    > driver.py 158
    > utils.py 159
> error-analysis 160
> examples 161
> figures 162
> notebooks 163
> results 164
> scripts 165
Structure
Bookmarks
Debug: run
Version Control TODO Problems Debug Python Packages Python Console Terminal Services Event Log
149:78 LF UTF-8 4 spaces Python 3.8

```

```

def process_new_case(self, ambulances, case: Case, current_time: datetime):
    # Select an ambulance
    selected_ambulance = self.select_ambulance(ambulances, case, current_time)
    selected_ambulance.deployed = True

    self.print("Selected ambulance: {}".format(selected_ambulance.id))

    # Add new case to ongoing cases
    case_event_iterator = case.iterator[selected_ambulance, current_time]
    case.next_event = next(case.event_iterator)
    case.event_finish_datetime = current_time + case.next_event.duration

    # TODO quite a bit of repeated code for self.print statements
    self.print("Started new event: {}".format(case.next_event.event_type.value))
    self.print("Destination: {}, {}".format(case.next_event.destination.latitude, case.next_event.destination.longitude))
    self.print("Duration: {}".format(case.next_event.duration))
    err = "{}".format(round(case.next_event.error, 2)) if case.next_event.error else None
    self.print("Distance Accuracy: {}".format(err))

    case_record = CaseRecord(case=case,
                            ambulance=selected_ambulance,
                            start_time=current_time,
                            event_history=[case.next_event])

```

Figure 29: event_simulator.py source code.

Simulator process and output

Starting with the selection of the ambulance that will attend the new case.

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Algorithm > ems > simulators > event_simulator.py

Project

Algorithm /-Descargas/Alg 140
analysis 141
configurations 142
docker 143
ems 144
algorithms 145
analysis 146
config_reader 147
datasets 148
generators 149
models 150
scenarios 151
simulators 152
init.py 153
event_simulator.py 154
simulator.py 155
triggers 156
init.py 157
driver.py 158
utils.py 159
error-analysis 160
examples 161
figures 162
notebooks 163
results 164
scripts

Run Simulation example.yaml random_case_set.py event_simulator.py simulator.py

```
def process_new_case(self, ambulances, case: Case, current_time: datetime):  
    # Select an ambulance  
    selected_ambulance = self.select_ambulance(ambulances, case, current_time)  
    selected_ambulance.deployed = True  
  
    self.print('Selected ambulance: {}'.format(selected_ambulance.id))  
  
    # Add new case to ongoing cases  
    case_event_iterator = case.iterator(selected_ambulance, current_time)  
    case_next_event = next(case_event_iterator)  
    case_event_finish_datetime = current_time + case_next_event.duration  
  
    # TODO quite a bit of repeated code for self.print statements  
    self.print("Started new event: {}".format(case_next_event.event_type.value))  
    self.print("Destination: {}, {}".format(case_next_event.destination.latitude, case_next_event.destination.longitude))  
    self.print("Duration: {}".format(case_next_event.duration))  
    err = "{}".format(round(case_next_event.error, 2)) if case_next_event.error else None  
    self.print("Distance Accuracy: {}".format(err))  
  
    case_record = CaseRecord(case=case,  
                             ambulance=selected_ambulance,  
                             start_time=current_time,  
                             event_history=[case_next_event])
```

EventDispatcherSimulator | process_new_case()

Debug: run

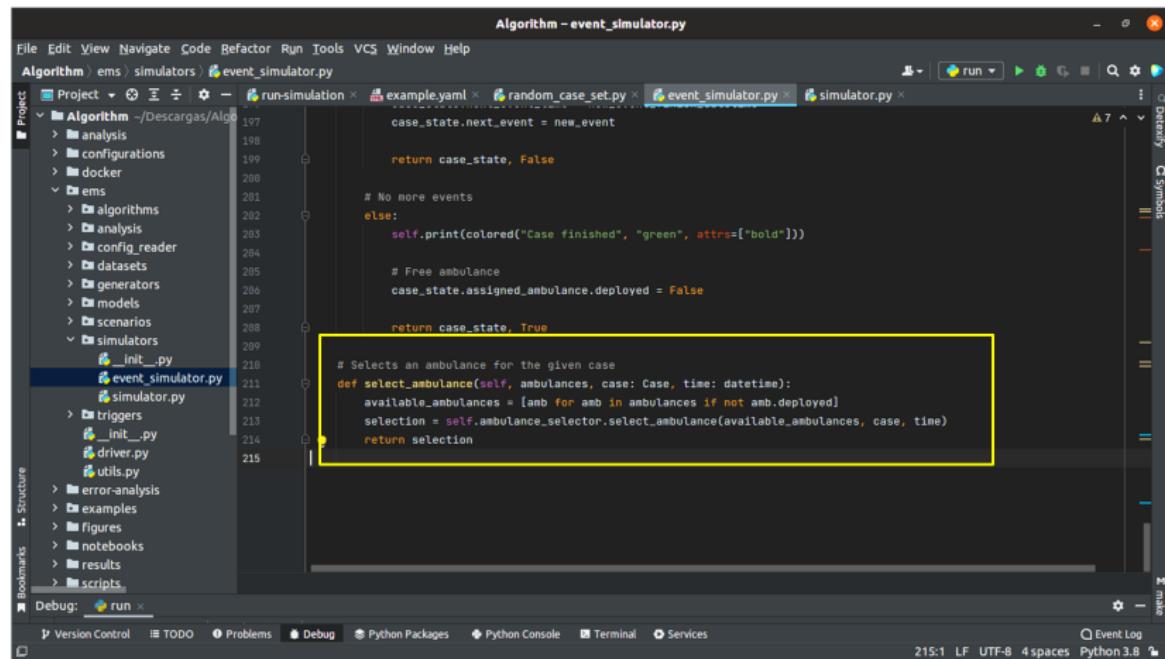
Version Control TODO Problems Debug Python Packages Python Console Terminal Services

149:78 LF UTF-8 4 spaces Python 3.8

Figure 29: event_simulator.py source code.

Simulator process and output

The `selec_ambulance` function is invoked, which is responsible for initiating this process.



```
Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm /ems > simulators > event_simulator.py
Project ▾ Algorithm -/Descargas/Algo
  > analysis
  > configurations
  > docker
  > ems
    > algorithms
    > analysis
    > config_reader
    > datasets
    > generators
    > models
    > scenarios
    > simulators
      > __init__.py
      > event_simulator.py
      > simulator.py
    > triggers
      > __init__.py
      > driver.py
      > utils.py
  > error-analysis
  > examples
  > figures
  > notebooks
  > results
  > scripts
Structure
Bookmarks
Debug: run ×
Version Control TODO Problems Debug Python Packages Python Console Terminal Services
Event Log
215:1 LF UTF-8 4 spaces Python 3.8 %
  197     case_state.next_event = new_event
  198
  199     return case_state, False
  200
  201     # No more events
  202     else:
  203         self.print(colored("Case finished", "green", attrs=["bold"]))
  204
  205         # Free ambulance
  206         case_state.assigned_ambulance.deployed = False
  207
  208         return case_state, True
  209
  210     # Selects an ambulance for the given case
  211     def select_ambulance(self, ambulances, case: Case, time: datetime):
  212         available_ambulances = [amb for amb in ambulances if not amb.deployed]
  213         selection = self.ambulance_selector.select_ambulance(available_ambulances, case, time)
  214
  215         return selection
```

Figure 29: event_simulator.py source code.

Simulator process and output

In turn is called the block that is responsible for carry out the process.

```

Project: Algorithm ~/Descargas/Algorithm
File Edit View Navigate ⌘
Algorithm ems > simulator
  Project ~ / Descargas/Algorithm
    - Algorithm ~/Descargas/Algorithm
      > analysis
      > configurations
        > coverage-depletion
        > debug
        > scenarios
        > weeks
          example.yaml
          example_scenario.yaml
          minimal.yaml
          simple.yaml
          simple-case-gen.yaml
        __init__.py
        event_simulator.py
        simulator.py
      triggers
      __init__.py
      driver.py
      utils.py
    error-analysis
    examples
    figures
    notebooks
    results
    scripts
  Debug: run ×
  Version Control TODO Problems Debug Python Packages Python Console Terminal Services
  215:1 LF UTF-8 4 spaces Python 3.8
  Event Log
  2

```

```

simulator: # Simulator
  class: ems.simulators.event_simulator.EventDispatcherSimulator
  debug: True
  ambulances: $ambulances
  cases: $cases
  ambulance_selector:
    class: ems.algorithms.selection.dispatch_fastest.BestTravelTime
    travel_times: $tt
  metric_aggregator: $metric_aggregator

```

```

# Selects an ambulance for the given case
def select_ambulance(self, ambulances, case: Case, time: datetime):
    available_ambulances = [amb for amb in ambulances if not amb.deployed]
    selection = self.ambulance_selector.select_ambulance(available_ambulances, case, time)
    return selection

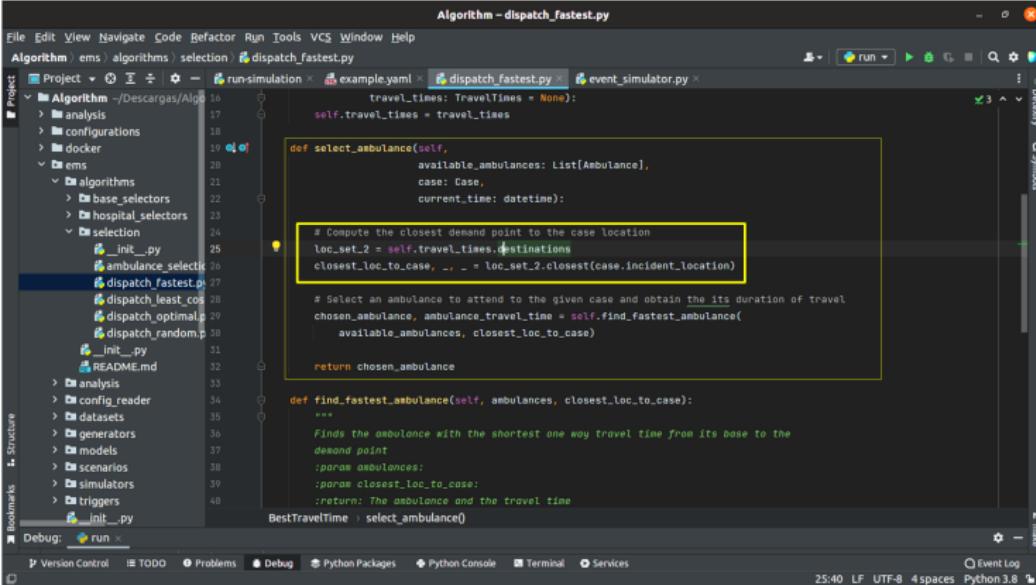
```

Figure 29: event_simulator.py source code.

Simulator process and output

Once inside the file.

1. The new case is located in one of the demand points.



```

Algorithm - dispatch_fastest.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm (ems) algorithms > selection dispatch_fastest.py example.yaml event_simulator.py
Project Project run-simulation travel_times: TravelTimes = None
> analysis self.travel_times = travel_times
> configurations
> docker
> ems
> ems
> algorithms
> base_selectors
> hospital_selectors
> selection
> _init_.py
> ambulance_selection.py
> dispatch_fastest.py
> dispatch_least_cost.py
> dispatch_optimal.py
> dispatch_random.py
> README.md
> analysis
> config_reader
> datasets
> generators
> models
> scenarios
> simulators
> triggers
> _init_.py

def select_ambulance(self,
                     available_ambulances: List[Ambulance],
                     case: Case,
                     current_time: datetime):
    # Compute the closest demand point to the case location
    loc_set_2 = self.travel_times.destinations
    closest_loc_to_case, _ = loc_set_2.closest(case.incident_location)

    # Select an ambulance to attend to the given case and obtain its duration of travel
    chosen_ambulance, ambulance_travel_time = self.find_fastest_ambulance(
        available_ambulances, closest_loc_to_case)

    return chosen_ambulance

def find_fastest_ambulance(self, ambulances, closest_loc_to_case):
    ...
    Finds the ambulance with the shortest one way travel time from its base to the
    demand point
    :param ambulances:
    :param closest_loc_to_case:
    :return: The ambulance and the travel time
BestTravelTime > select_ambulance()

```

Figure 30: dispatch_fastest.py source code.

Simulator process and output

2. Check the distance in the travel time matrix of each ambulance to the demand point, until you find the ambulance that will arrive in the shortest possible time.

The screenshot shows a Python development environment with the following details:

- Project:** Algorithm - dispatch_fastest.py
- File:** dispatch_fastest.py
- Code:** The code implements an ambulance selector. It defines a `select_ambulance` method that finds the closest demand point to a given case location and then selects the fastest ambulance from a list of available ones. A helper function `find_fastest_ambulance` is used to find the ambulance with the shortest travel time from its base to the demand point.

```
def select_ambulance(self, available_ambulances: List[Ambulance], case: Case, current_time: datetime):
    # Compute the closest demand point to the case location
    loc_set_2 = self.travel_times.destinations
    closest_loc_to_case, _ = loc_set_2.closest(case.incident_location)

    # Select an ambulance to attend to the given case and obtain the its duration of travel
    chosen_ambulance, ambulance_travel_time = self.find_fastest_ambulance(
        available_ambulances, closest_loc_to_case)

    return chosen_ambulance

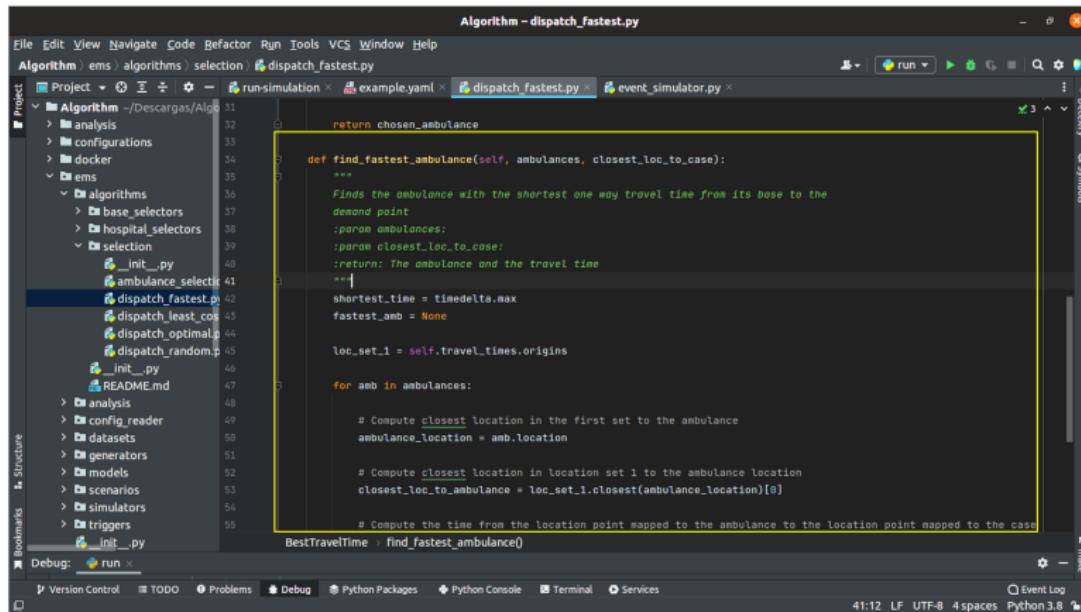
def find_fastest_ambulance(self, ambulances, closest_loc_to_case):
    """
    Finds the ambulance with the shortest one way travel time from its base to the
    demand point
    :param ambulances:
    :param closest_loc_to_case:
    :return: The ambulance and the travel time
    BestTravelTime > select_ambulance()
```

- IDE Features:** The code is syntax-highlighted with orange for strings and blue for functions. A yellow arrow points from the `self` parameter in the `find_fastest_ambulance` call to the `self` parameter in the same function's definition below it. The interface includes a top navigation bar with File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and a search bar. On the left, there's a sidebar for Project, Structure, and Bookmarks. At the bottom, there are tabs for Version Control, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, and a status bar showing the current time (25:40), file format (LF), encoding (UTF-8), and number of spaces (4 spaces).

Figure 30: dispatch_fastest.py source code.

Simulator process and output

2. Check the distance in the travel time matrix of each ambulance to the demand point, until you find the ambulance that will arrive in the shortest possible time.



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Algorithm - dispatch_fastest.py
- File:** dispatch_fastest.py
- Code Content:**

```

Algorithm - dispatch_fastest.py
-----
return chosen_ambulance

def find_fastest_ambulance(self, ambulances, closest_loc_to_case):
    """
    Finds the ambulance with the shortest one way travel time from its base to the
    demand point
    :param ambulances:
    :param closest_loc_to_case:
    :return: The ambulance and the travel time
    """
    shortest_time = timedelta.max
    fastest_amb = None

    loc_set_1 = self.travel_times.origins

    for amb in ambulances:
        # Compute closest location in the first set to the ambulance
        ambulance_location = amb.location

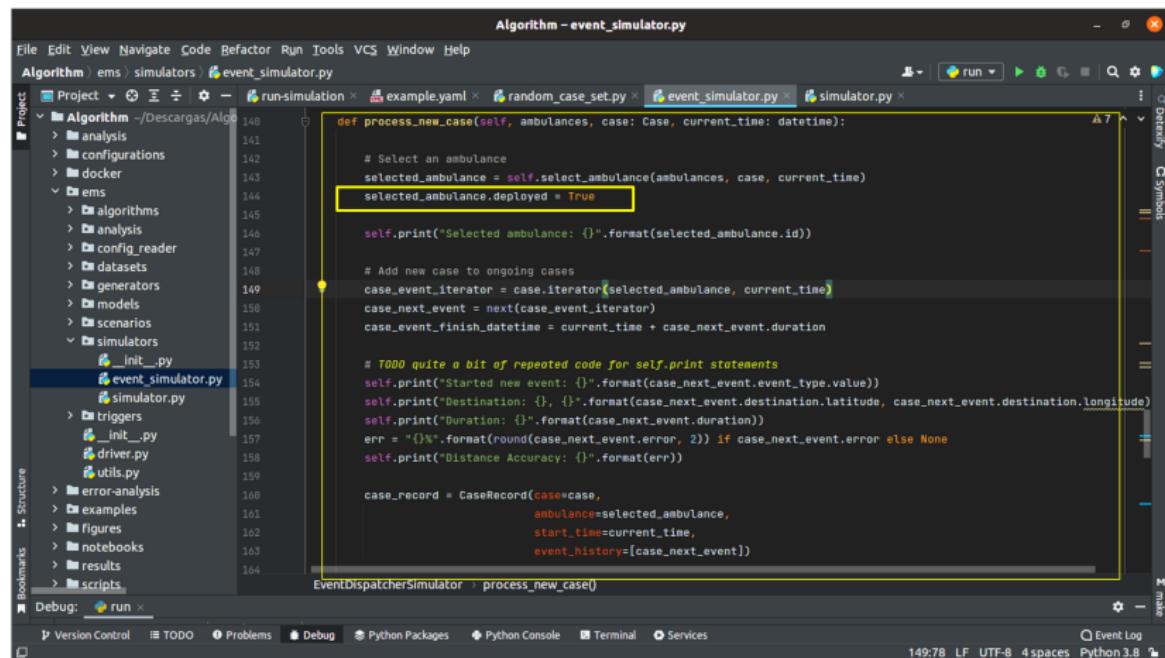
        # Compute closest location in location set 1 to the ambulance location
        closest_loc_to_ambulance = loc_set_1.closest(ambulance_location)[0]

        # Compute the time from the location point mapped to the ambulance to the location point mapped to the case
        BestTravelTime = find_fastest_ambulance()
    
```
- Toolbars and Menus:** Standard PyCharm menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Bottom Status Bar:** Version Control, TODO, Problems, Debug, Python Packages, Python Console, Terminal, Services, Event Log, 41:12 LF, UTF-8, 4 spaces, Python 3.8.

Figure 30: dispatch_fastest.py source code.

Simulator process and output

Continuing with the new case, once the ambulance is selected, it will be set as occupied.



```

Algorithm -event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm /ems/simulators/event_simulator.py
Project ▾ Algorithm -/Descargas/Algo
  ▾ analysis
  ▾ configurations
  ▾ docker
  ▾ ems
    ▾ algorithms
    ▾ analysis
    ▾ config_reader
    ▾ datasets
    ▾ generators
    ▾ models
    ▾ scenarios
    ▾ simulators
      ▾ __init__.py
      ▾ event_simulator.py
      ▾ simulator.py
    ▾ triggers
      ▾ __init__.py
      ▾ driver.py
      ▾ utils.py
  ▾ error-analysis
  ▾ examples
  ▾ figures
  ▾ notebooks
  ▾ results
  ▾ scripts
  ▾ run-simulation
  ▾ example.yaml
  ▾ random_case_set.py
  ▾ event_simulator.py
  ▾ simulator.py
  ▾ EventDispatcherSimulator
    ▾ process_new_case()
  ▾ make

def process_new_case(self, ambulances, case: Case, current_time: datetime):
    # Select an ambulance
    selected_ambulance = self.select_ambulance(ambulances, case, current_time)
    selected_ambulance.deployed = True

    self.print("Selected ambulance: {}".format(selected_ambulance.id))

    # Add new case to ongoing cases
    case_event_iterator = case.iterator(selected_ambulance, current_time)
    case_next_event = next(case_event_iterator)
    case_event_finish_datetime = current_time + case_next_event.duration

    # TODO quite a bit of repeated code for self.print statements
    self.print("Started new event: {}".format(case_next_event.event_type.value))
    self.print("Destination: {}, {}".format(case_next_event.destination.latitude, case_next_event.destination.longitude))
    self.print("Duration: {}".format(round(case_next_event.duration)))
    err = "{}".format(round(case_next_event.error, 2)) if case_next_event.error else None
    self.print("Distance Accuracy: {}".format(err))

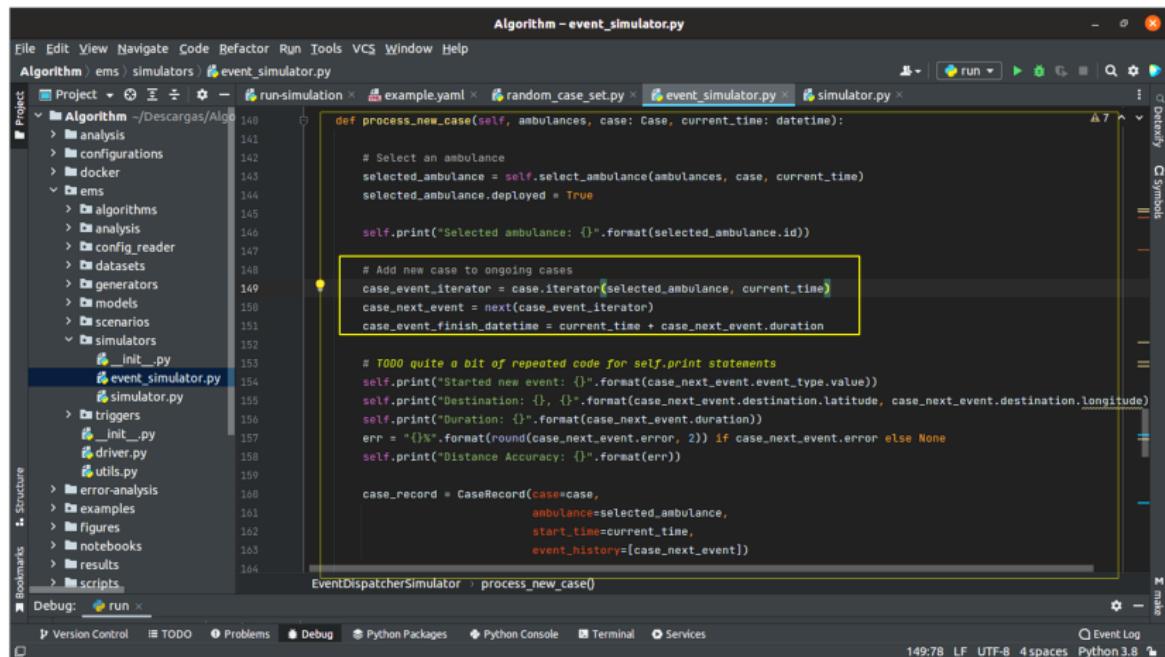
    case_record = CaseRecord(case=case,
                             ambulance=selected_ambulance,
                             start_time=current_time,
                             event_history=[case_next_event])

```

Figure 31: event_simulator.py source code

Simulator process and output

The first event generated is: **Arrive to incident**



```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm / ems / simulators / event_simulator.py
Project run-simulation example.yaml random_case_set.py event_simulator.py simulator.py
Algorithm / Descargas / Algo
> analysis
> configurations
> docker
> ems
> ems / algorithms
> analysis
> config_reader
> datasets
> generators
> models
> scenarios
> simulators
> _init_.py
> event_simulator.py
> simulator.py
> triggers
> _init_.py
> driver.py
> utils.py
> error-analysis
> examples
> figures
> notebooks
> results
> scripts
Debug: run
Version Control TODO Problems Debug Python Packages Python Console Terminal Services
Event Log 149:78 LF UTF-8 4 spaces Python 3.8
def process_new_case(self, ambulances, case: Case, current_time: datetime):
    # Select an ambulance
    selected_ambulance = self.select_ambulance(ambulances, case, current_time)
    selected_ambulance.deployed = True

    self.print("Selected ambulance: {}".format(selected_ambulance.id))

    # Add new case to ongoing cases
    case.event_iterator = case.iterator(selected_ambulance, current_time)
    case.next_event = next(case.event_iterator)
    case.event_finish_datetime = current_time + case.next_event.duration

    # TODO quite a bit of repeated code for self.print statements
    self.print("Started new event: {}".format(case.next_event.event_type.value))
    self.print("Destination: {}, {}".format(case.next_event.destination.latitude, case.next_event.destination.longitude))
    self.print("Duration: {}".format(case.next_event.duration))
    err = "{}".format(round(case.next_event.error, 2)) if case.next_event.error else None
    self.print("Distance Accuracy: {}".format(err))

    case_record = CaseRecord(case=case,
                             ambulance=selected_ambulance,
                             start_time=current_time,
                             event_history=[case.next_event])

```

Figure 31: event_simulator.py source code

Simulator process and output

The travel time from the base to the demand point is obtained from the previously saved matrix.

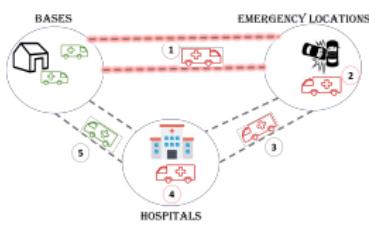


Figure 32: Case events

```

vertices_longitude: [-117.125500, -116.870454, -116.82, -116.907161, -117.125475]
event_generator:
  class: ems.generators.event.event_generator.EventGenerator
  travel_duration_generator: # Travel times between events estimated using travel time
    class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
    travel_times: $tt
    epsilon: 0.001
  incident_duration_generator: # Duration at incident drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_duration_generator: # Duration at hospital drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_selector: # Fastest hospital chosen
    class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
    hospital_set: $hospitals
    travel_times: $tt

```

Figure 33: example.yaml source code

Simulator process and output

The duration at the place of the new case is randomly generated between 5 and 10 minutes.

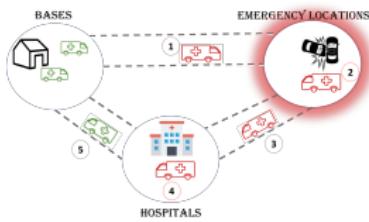


Figure 34: Case events

```
vertices_longitude: [-117.125500, -116.870454, -116.82, -116.907161, -117.125475]
event_generator:
  class: ems.generators.event.event_generator.EventGenerator
  travel_duration_generator: # Travel times between events estimated using travel time generator
    class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
    travel_times: $tt
    epsilon: 0.001
  incident_duration_generator: # Duration at incident drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_duration_generator: # Duration at hospital drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_selector: # Fastest hospital chosen
    class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
    hospital_set: $hospitals
    travel_times: $tt
```

Figure 35: example.yaml source code

Simulator process and output

The duration in the hospital is randomly generated between 5 and 10 minutes.

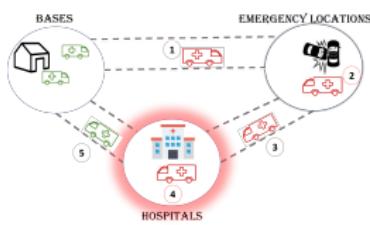


Figure 36: Case events

```

vertices_longitude: [-117.125500, -116.870454, -116.82, -116.907161, -117.125475]
event_generator:
  class: ems.generators.event.event_generator.EventGenerator
  travel_duration_generator: # Travel times between events estimated using travel time
    class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
    travel_times: $tt
    epsilon: 0.001
  incident_duration_generator: # Duration at incident drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_duration_generator: # Duration at hospital drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_selector: # Fastest hospital chosen
    class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
    hospital_set: $hospitals
    travel_times: $tt

```

Figure 37: example.yaml source code

Simulator process and output

The selection of the hospital is according to the one located closest to the point of demand.

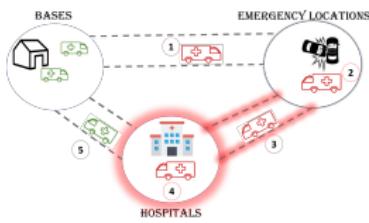


Figure 38: Case events

```

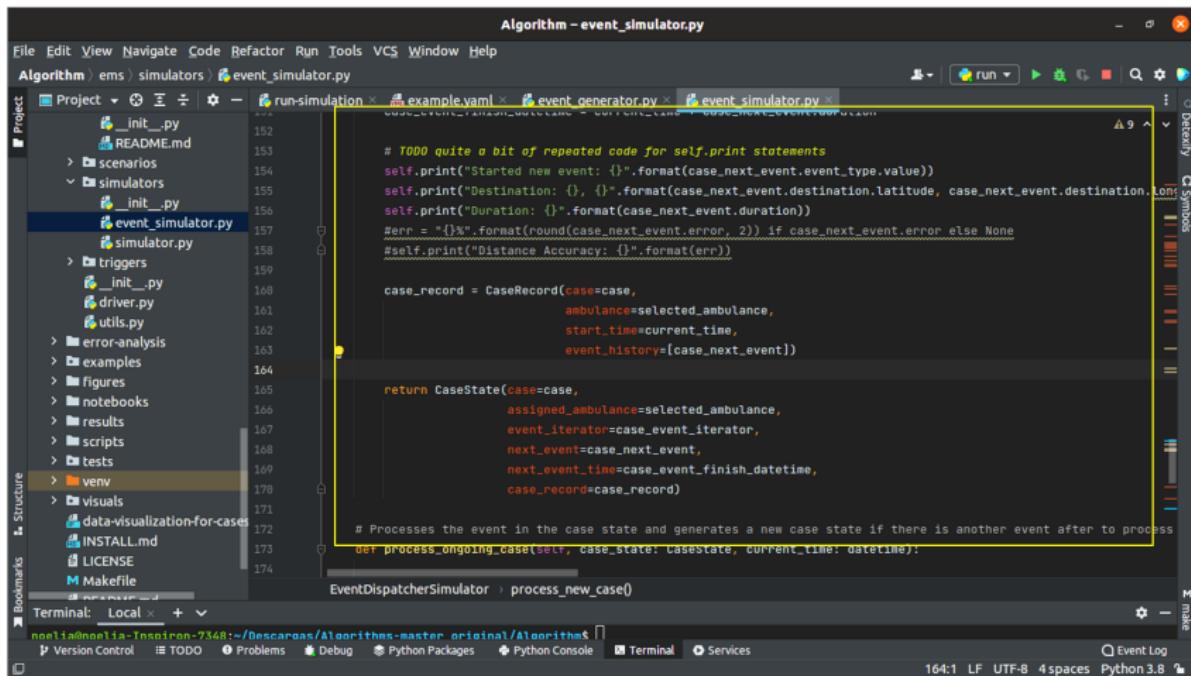
vertices_longitude: [117.123500, 116.870454, 116.82, 116.907161, 117.123475]
event_generator:
  class: ems.generators.event.event_generator.EventGenerator
  travel_duration_generator: # Travel times between events estimated using travel time
    class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
    travel_times: $tt
    epsilon: 0.001
  incident_duration_generator: # Duration at incident drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
  hospital_duration_generator: # Duration at hospital drawn from uniform distribution
    class: ems.generators.duration.random_duration.RandomDurationGenerator
    lower_bound: 5
    upper_bound: 10
hospital_selector: # Fastest hospital chosen
  class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
  hospital_set: $hospitals
  travel_times: $tt

```

Figure 39: example.yaml source code

Simulator process and output

Once the first event has occurred, the information is saved.



```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > ems > simulators > event_simulator.py
Project Structure Bookmarks Terminal: Local
event_simulator.py
# TODO quite a bit of repeated code for self.print statements
self.print("Started new event: {}".format(case_next_event.event_type.value))
self.print("Destination: {}, {}".format(case_next_event.destination.latitude, case_next_event.destination.longitude))
self.print("Duration: {}".format(case_next_event.duration))
#err = "{}".format(round(case_next_event.error, 2)) if case_next_event.error else None
#self.print("Distance Accuracy: {}".format(err))

case_record = CaseRecord(case=case,
                         assigned_ambulance=selected_ambulance,
                         start_time=current_time,
                         event_history=[case_next_event])

return CaseState(case=case,
                 assigned_ambulance=selected_ambulance,
                 event_iterator=case_event_iterator,
                 next_event=case_next_event,
                 next_event_time=case_event_finish_datetime,
                 case_record=case_record)

# Processes the event in the case state and generates a new case state if there is another event after to process
def process_onGoing_case(self, case_state: Casestate, current_time: datetime):
    # ...
EventDispatcherSimulator > process_new_case()
# Process the event in the case state and generates a new case state if there is another event after to process
def process_new_case(self, case_state: Casestate, current_time: datetime):
    # ...

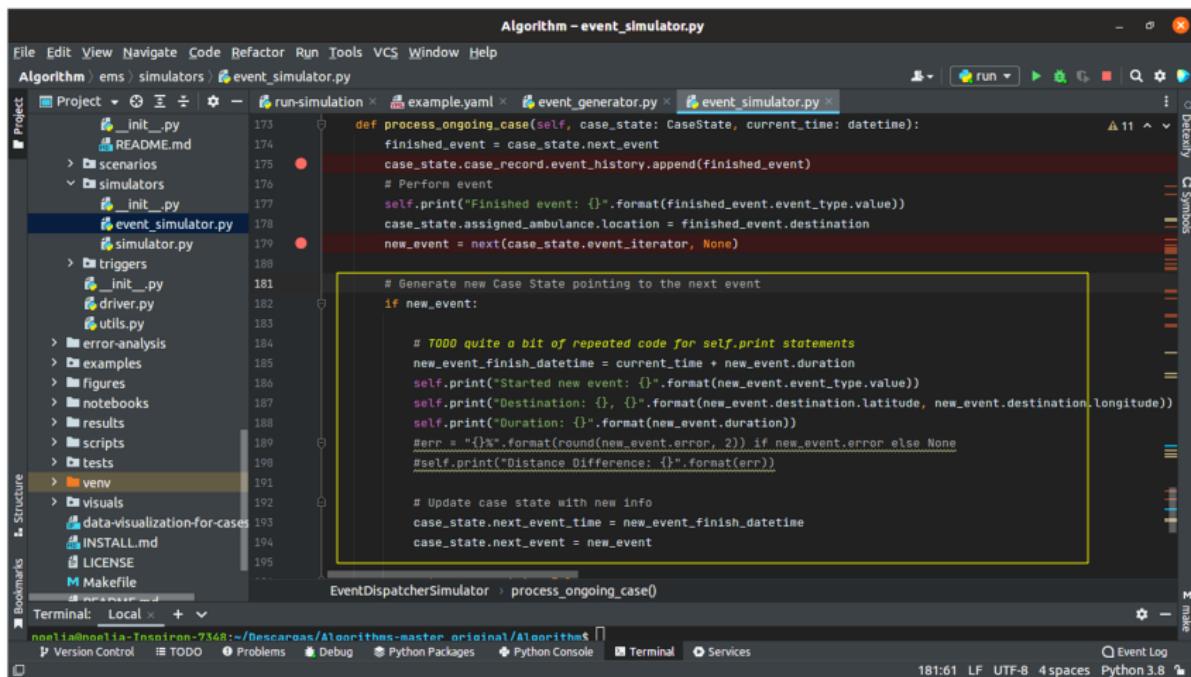
```

The screenshot shows a code editor with the file `event_simulator.py` open. The code is part of a larger project structure. A yellow box highlights the `process_new_case` function, which is responsible for processing events in the case state and generating a new case state if there is another event after. The code uses `self.print` statements to log information about the event type, destination, duration, and error. It also creates a `CaseRecord` and returns a `CaseState`. The function `process_onGoing_case` is also visible, which processes the event in the case state and generates a new case state if there is another event after to process.

Figure 40: `event_simulator.py` source code

Simulator process and output

Continuing this case, if it is the last event, terminates the case.



```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm > ems > simulators > event_simulator.py
Project run-simulation example.yaml event_generator.py event_simulator.py
event_simulator.py
173     def process_ongoing_case(self, case_state: CaseState, current_time: datetime):
174         finished_event = case_state.next_event
175         case_state.case_record.event_history.append(finished_event)
176         # Perform event
177         self.print("Finished event: {}".format(finished_event.event_type.value))
178         case_state.assigned_ambulance.location = finished_event.destination
179         new_event = next(case_state.event_iterator, None)

# Generate new Case State pointing to the next event
if new_event:

    # TODO quite a bit of repeated code for self.print statements
    new_event_finish_datetime = current_time + new_event.duration
    self.print("Started new event: {}".format(new_event.event_type.value))
    self.print("Destination: {}, {}".format(new_event.destination.latitude, new_event.destination.longitude))
    self.print("Duration: {}".format(new_event.duration))
    err = "{}".format(round(new_event.error, 2)) if new_event.error else None
    self.print("Distance Difference: {}".format(err))

    # Update case state with new info
    case_state.next_event_time = new_event_finish_datetime
    case_state.next_event = new_event

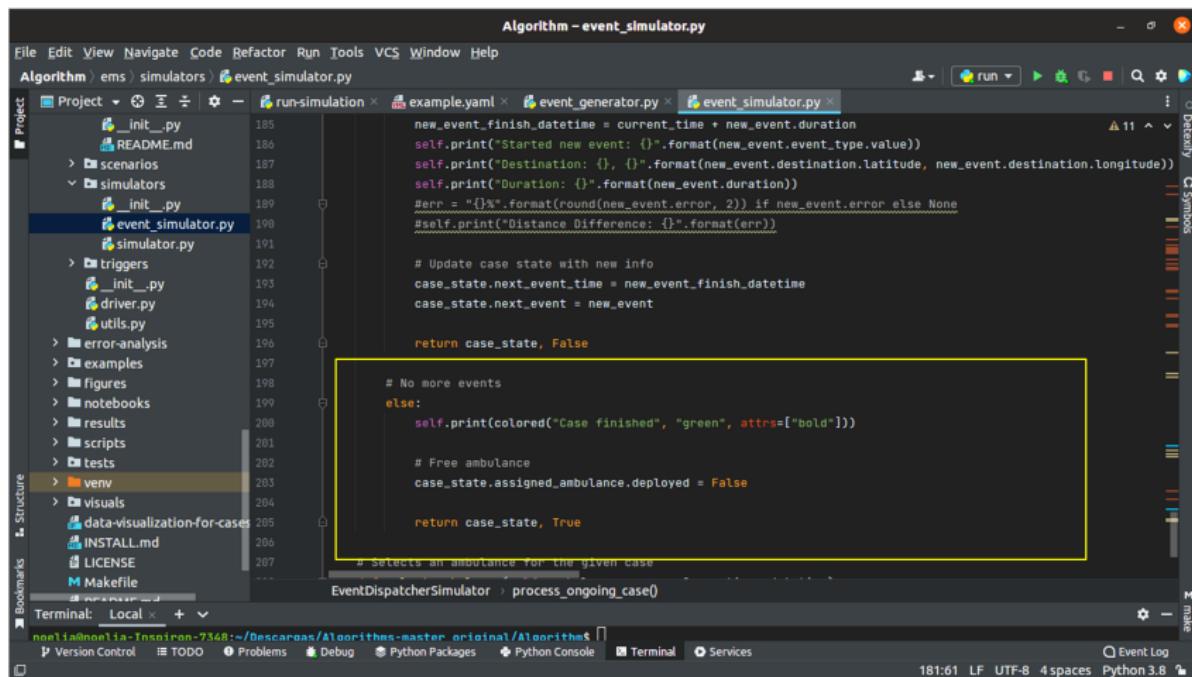
EventDispatcherSimulator > process_ongoing_case()

```

Figure 40: event_simulator.py source code

Simulator process and output

Continuing this case, if it is the last event, terminates the case.



```

Algorithm - event_simulator.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithm\ems\simulators>event_simulator.py
Project < run-simulation > example.yaml > event_generator.py > event_simulator.py
 185     new_event_finish_datetime = current_time + new_event.duration
 186     self.print("Started new event: {}.".format(new_event.event_type.value))
 187     self.print("Destination: {}, {}".format(new_event.destination.latitude, new_event.destination.longitude))
 188     self.print("Duration: {}.".format(new_event.duration))
 189     #err = "{}%".format(round(new_event.error, 2)) if new_event.error else None
 190     #self.print("Distance Difference: {}.".format(err))
 191
 192     # Update case state with new info
 193     case_state.next_event_time = new_event.finish_datetime
 194     case_state.next_event = new_event
 195
 196     return case_state, False
 197
 198     # No more events
 199     else:
 200         self.print(colored("Case finished", "green", attrs=["bold"]))
 201
 202         # Free ambulance
 203         case_state.assigned_ambulance.deployed = False
 204
 205         return case_state, True
 206
 207     # Selects an ambulance for the given case
 208
 209 EventDispatcherSimulator > process_ongoing_case()

```

The screenshot shows a code editor with the file `event_simulator.py` open. The code implements a simulator for an ambulance case. It processes events by updating the case state and printing progress messages. If there are no more events, it prints a success message and sets the ambulance's deployed status to False. A yellow box highlights the final logic in the `process_ongoing_case()` function, specifically the part where it checks if there are no more events and then returns the case state and a boolean value indicating the case is finished.

Figure 40: `event_simulator.py` source code

Simulator process and output

Another important point to highlight is that there is a possibility that a case may not be attended because ambulances are not available, therefore, the case remains pending until it can be addressed.

```

Algorithm - event_simulator.py
File Edit View Navigate Code Befactor Run Tools VCS Window Help
Algorithm.ems > simulators > event_simulator.py
Project > run-simulation > example.yaml > event_generator.py > event_simulator.py
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
# Process a new case
if available_ambulances:
    self.print(colored("Processing new case: {}.".format(next_case.id), "green", attrs=[ "bold" ]))
    case_state_to_add = self.process_new_case(ambulances, next_case, current_time)
    bisect.insort(left(ongoing_case_states, case_state_to_add))

# Delay a case
else:
    self.print(colored("New case arrived but no available ambulance; Case pending".format(), "red"))
    pending_cases.append(next_case)

# Prepare the next case
next_case = next(case_iterator, None)

# Process an ongoing case event
else:

    next_ongoing_case_state = ongoing_case_states.pop(0) # next_ongoing_case_state: <ems.simulators.event_
    current_time = next_ongoing_case_state.next_event_time

    self.print(colored("Current Time: {}".format(current_time), "cyan", attrs=[ "bold" ]))
    self.print(colored("Processing ongoing case: {}.".format(next_ongoing_case_state.case_id),
                      "magenta"))

```

Figure 40: event_simulator.py source code

Simulator process and output

From this block you enter the module to calculate the percentage of coverage, within the minimum established time.

The screenshot shows a software interface with a menu bar at the top: File, Edit, View, Navigate, Code, Bafactor, Run, Tools, VCS, Window, Help. Below the menu is a toolbar with icons for run, debug, and search. The main area is titled "Algorithm - example.yaml". On the left is a "Project" sidebar with a tree view containing "Algorithm", "analysis", "configurations", "docker", "ems", "error-analysis", "examples", "figures", "notebooks", "results", "scripts", "tests", "venv", "visuals", "data-visualization-for-ca", "INSTALL.md", "LICENSE", "Makefile", "README.md", "requirements.txt", "run-simulation", "External Libraries", and "Scratches and Consoles". The "example.yaml" file is open in the center, showing YAML configuration code. A yellow box highlights a section of the code starting with "metric_aggregator: # Computes and stores any amount of metrics during the simulation". The code defines various metrics and their configurations. At the bottom, there is a terminal tab labeled "Local" and a status bar showing "71:43 LF UTF-8 2 spaces No JSON schema Python 3.8".

```
lower_bound: 5
upper_bound: 10
hospital_selector: # Fastest hospital chosen
class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
hospital_set: $hospitals
travel_time: $tt

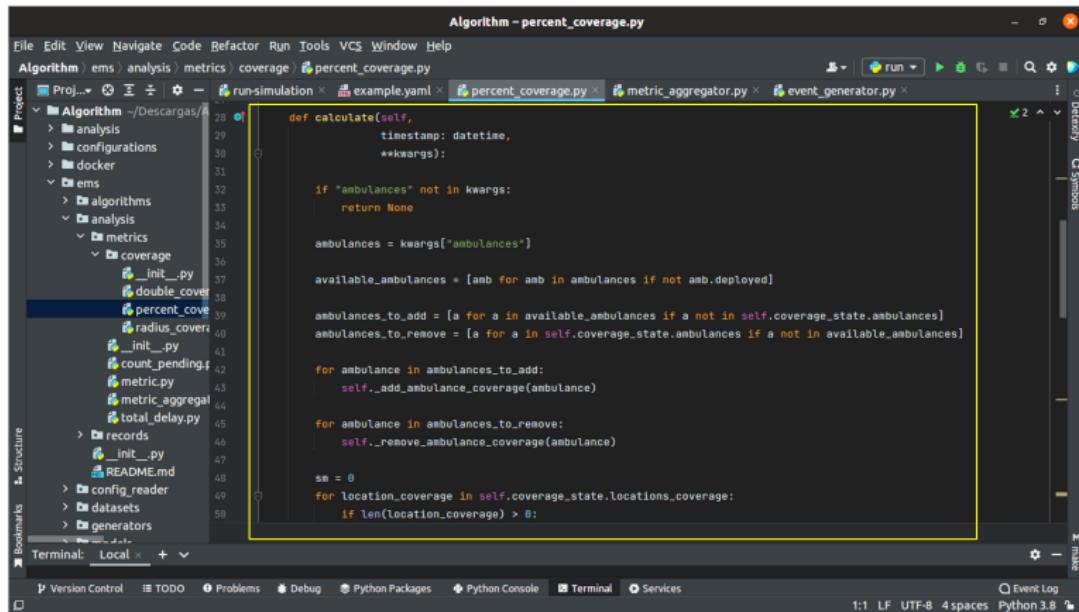
metric_aggregator: # Computes and stores any amount of metrics during the simulation
class: ems.analysis.metrics.metric_aggregator.MetricAggregator
metrics:
- class: ems.analysis.metrics.coverage.percent_coverage.PercentCoverage
r1: 600
demands: $demands
travel_time: $tt
- class: ems.analysis.metrics.total_delay.TotalDelay
- class: ems.analysis.metrics.count_pending.CountPending

simulator: # Simulator
class: ems.simulators.event_simulator.EventDispatcherSimulator
debug: True
ambulances: $ambulances
cases: $cases
ambulance_selector:
class: ems.algorithms.selection.dispatch_fastest.BestTravelTime
```

Figure 41: example.yaml source code

Simulator process and output

It is calculated for each event, this depends on how many ambulances are available and how many demand points are covered.



```
Algorithm – percent_coverage.py
File Edit View Navigate Code Befactor Run Tools VCS Window Help
Algorithm / ems / analysis / metrics / coverage / percent_coverage.py
Project: Algorithm -> Descargas/...
  > analysis
  > configurations
  > docker
  > ems
    > algorithms
      > analysis
        > metrics
          > coverage
            > __init__.py
            > double_cover
            > percent_cov
            > radius_cover
            > __init__.py
            > count_pending
            > metric.py
            > metric_aggregat
            > total_delay.py
        > records
          > __init__.py
        > README.md
    > config_reader
    > datasets
    > generators
run-simulation example.yaml percent_coverage.py metric_aggregator.py event_generator.py
def calculate(self,
              timestamp: datetime,
              **kwargs):
    if "ambulances" not in kwargs:
        return None

    ambulances = kwargs["ambulances"]

    available_ambulances = [amb for amb in ambulances if not amb.deployed]

    ambulances_to_add = [a for a in available_ambulances if a not in self.coverage_state.ambulances]
    ambulances_to_remove = [a for a in self.coverage_state.ambulances if a not in available_ambulances]

    for ambulance in ambulances_to_add:
        self._add_ambulance_coverage(ambulance)

    for ambulance in ambulances_to_remove:
        self._remove_ambulance_coverage(ambulance)

    sm = 0
    for location_coverage in self.coverage_state.locations_coverage:
        if len(location_coverage) > 0:
```

Figure 42: percent_coverage.py source code

Simulator process and output

Once the execution is completed, the results are obtained, which are the cases generated with their respective events and duration, as well as the percentage of coverage.

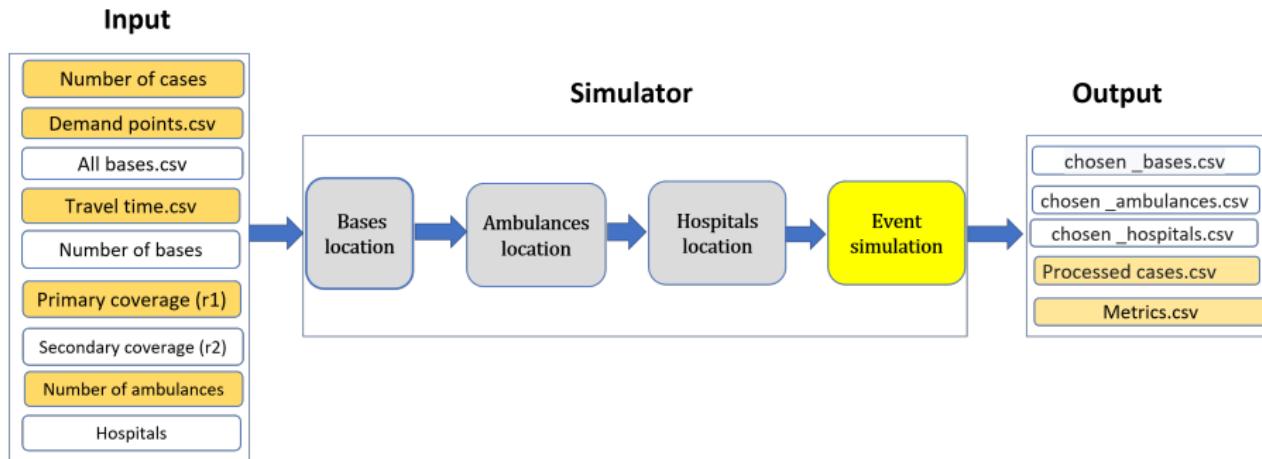


Figure 43: Simulator module.

Simulator process and output

The file contains the history of each case, the time taken for each event and the ambulance that attended the service.

A	B	C	D	E	F	G	H	I	J
id	date	latitude	longitude	priority	ambulance	start_time	TO INCIDENT	duration AT INCIDENT	duration TO HOSPITAL
2	2001-12-15 03:17:05.82330	32.492455248889	-116.90452135648	4	0	2001-12-15 03:17:05.82330	0 days 00:05:27	0 days 00:05:38	0 days 00:22:01
3	2001-12-15 03:19:09.37050	32.51932117419	-117.00165651897	4	1	2001-12-15 03:19:09.37050	0 days 00:06:32	0 days 00:08:46	0 days 00:09:24
4	2001-12-15 03:34:38.18260	32.470963940057	-116.95479433808	4	3	2001-12-15 03:34:38.18260	0 days 00:09:51	0 days 00:06:47	0 days 00:27:08
5	2001-12-15 03:35:39.50999	32.499295853542	-116.96736688059	3	5	2001-12-15 03:35:39.50999	0 days 00:02:29	0 days 00:05:52	0 days 00:18:33
6	2001-12-15 03:52:23.52454	32.5051438277	-116.99612325133	3	2	2001-12-15 03:52:23.52454	0 days 00:12:37	0 days 00:09:30	0 days 00:14:12
7	2001-12-15 03:53:54.31600	32.473242429891	-117.00800297838	4	4	2001-12-15 03:53:54.31600	0 days 00:05:11	0 days 00:07:00	0 days 00:15:55
8	2001-12-15 03:55:47.28490	32.516285293741	-117.02890675713	4	11	2001-12-15 03:55:47.28490	0 days 00:15:59	0 days 00:07:25	0 days 00:09:51
9	2001-12-15 04:07:28.14579	32.478889918221	-117.0214460685	4	1	2001-12-15 04:07:28.14579	0 days 00:09:41	0 days 00:06:58	0 days 00:18:16
10	2001-12-15 04:18:38.44480	32.506569243247	-117.05595658959	4	8	2001-12-15 04:18:38.44480	0 days 00:23:06	0 days 00:07:42	0 days 00:15:21
11	2001-12-15 04:38:43.30829	32.519133347492	-117.02784372881	2	5	2001-12-15 04:38:43.30829	0 days 00:17:27	0 days 00:09:39	0 days 00:08:43
12									
13									
14									

Figure 44: Processed file.

Simulator process and output

The file contains the current time of the event, the percentage of coverage of the demand points, whether it has taken more than one day to complete, and pending cases if any.

	A	B	C	D	E	F	G	H	I
1	timestamp	percent_coverage	total_delay	count_pending					
2	2001-12-15 03:17:05.823363	0.9444444444444444	0 days	0					
3	2001-12-15 03:19:09.370502	0.8333333333333333	0 days	0					
4	2001-12-15 03:22:32.823363	0.8333333333333333	0 days	0					
5	2001-12-15 03:25:41.370502	0.8333333333333333	0 days	0					
6	2001-12-15 03:28:10.823363	0.8333333333333333	0 days	4					
7	2001-12-15 03:34:27.370502	0.8333333333333333	0 days	0					
8	2001-12-15 03:34:38.182600	0.7222222222222222	0 days	0					
9	2001-12-15 03:35:39.509959	0.6666666666666667	0 days	0					
10	2001-12-15 03:38:08.509959	0.6666666666666667	0 days	0					

Figure 44: Metrics file.

Simulator input

The video shows the process of the simulator (original version).

[Watch video](#)

The screenshot shows the PyCharm IDE interface with the following details:

- Code Editor:** The main window displays a Python script named `run-simulation.py`. The code initializes configurations, reads user input, creates a driver object, and checks if the `figures` folder exists.
- Terminal:** Below the code editor, the terminal window shows the execution of the script. It prints the current time, processes an ongoing case, and handles a finished event by returning to the base. It also lists case, busy ambulances, ongoing cases, and pending cases.
- Status Bar:** The bottom status bar shows the current time as 10:32, file encoding as UTF-8, and Python version as 3.8.

6. Integration of the simulator module to the interactive system for ambulance dispatching in the city of Tijuana

Integration of the Simulator module

The system proposed to be developed presents significant advances in the problem addressed.

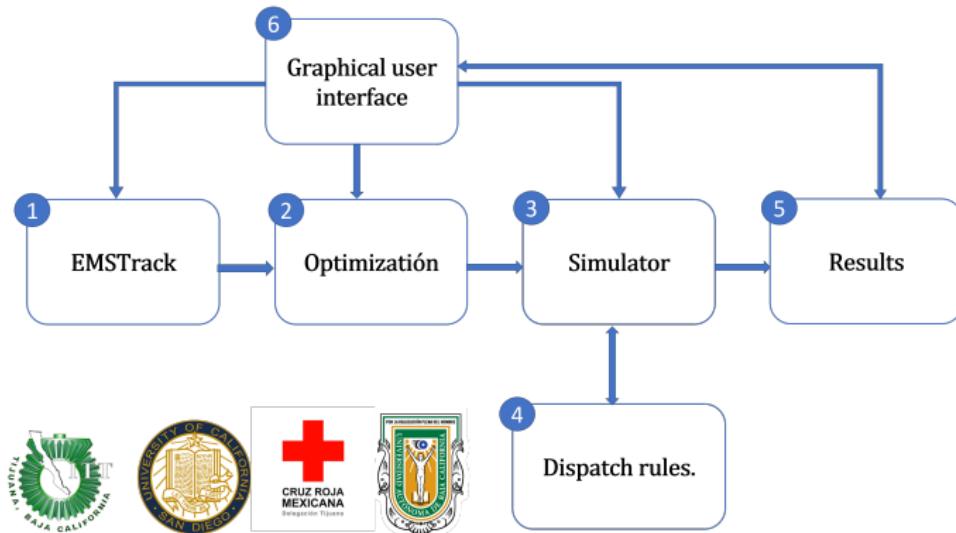


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

We have EMSTrack, a software module developed by UCSD, for ambulance management and tracking, which allows real-time geolocation of the units and allows detailed general call histories.

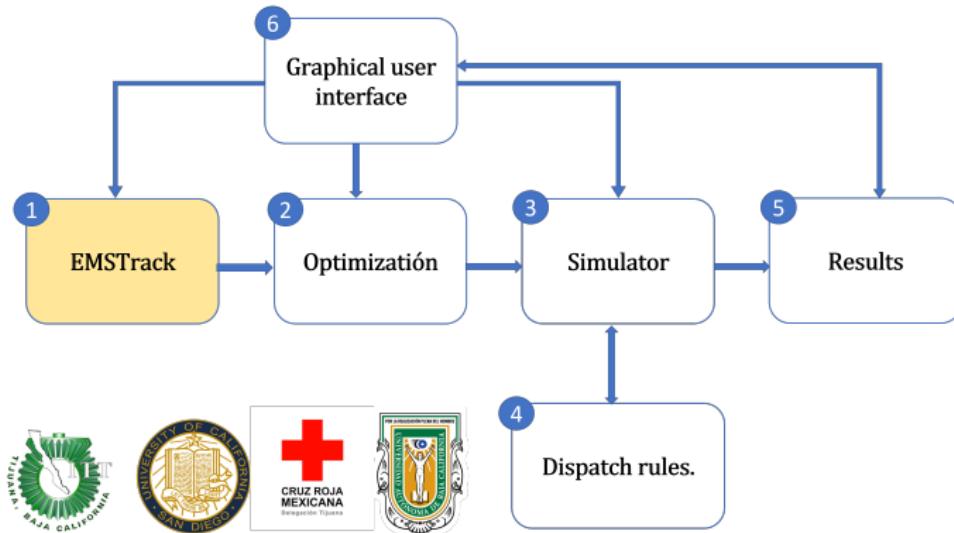


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Continuing with the next optimization module, which includes methods for analyzing and clustering the historical call data generated in module 1, as well as for modeling and optimizing problems and relocation problems based on the DSM model.

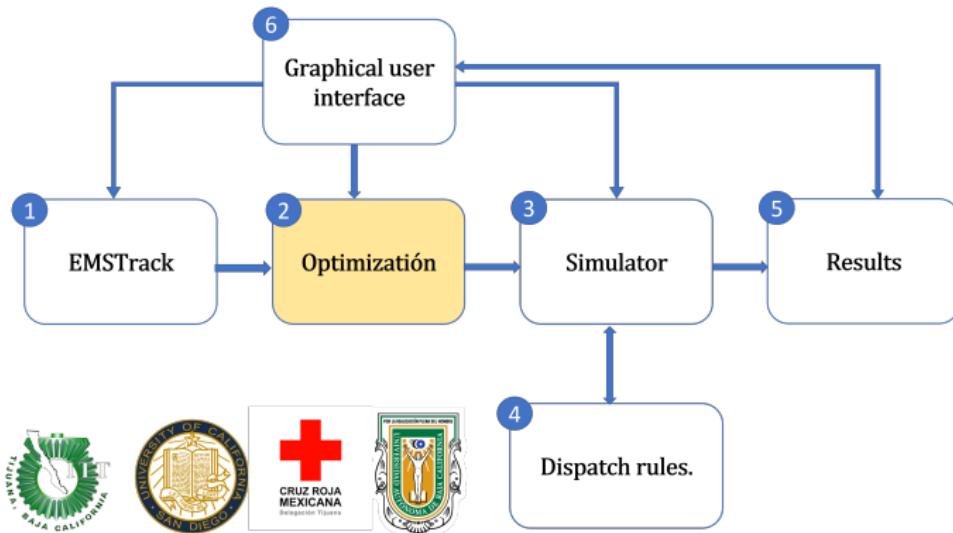


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

In module 3 we have the simulator that is based on state machines that allow us to evaluate different events, call frequency, ambulance location and response times.

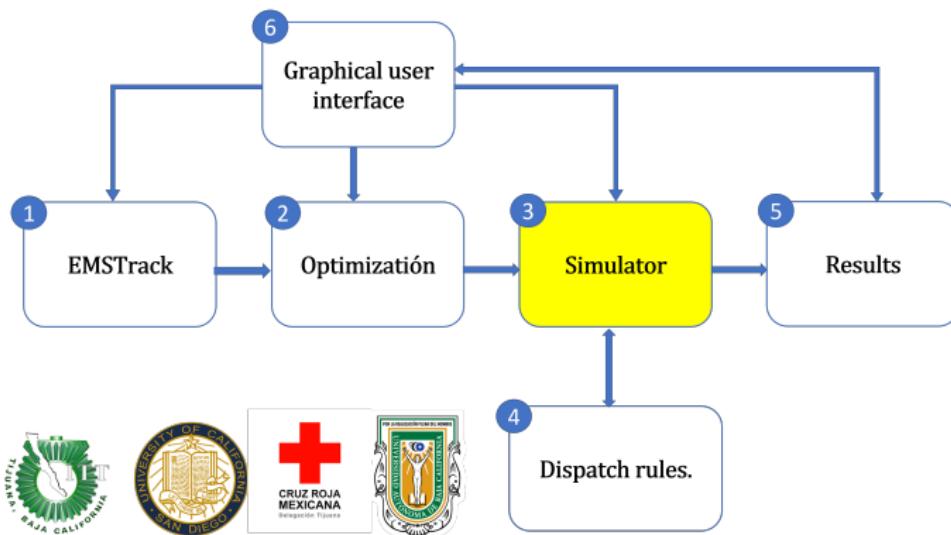


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Module 4 corresponds to the ambulance dispatch rules and is pending to be added.

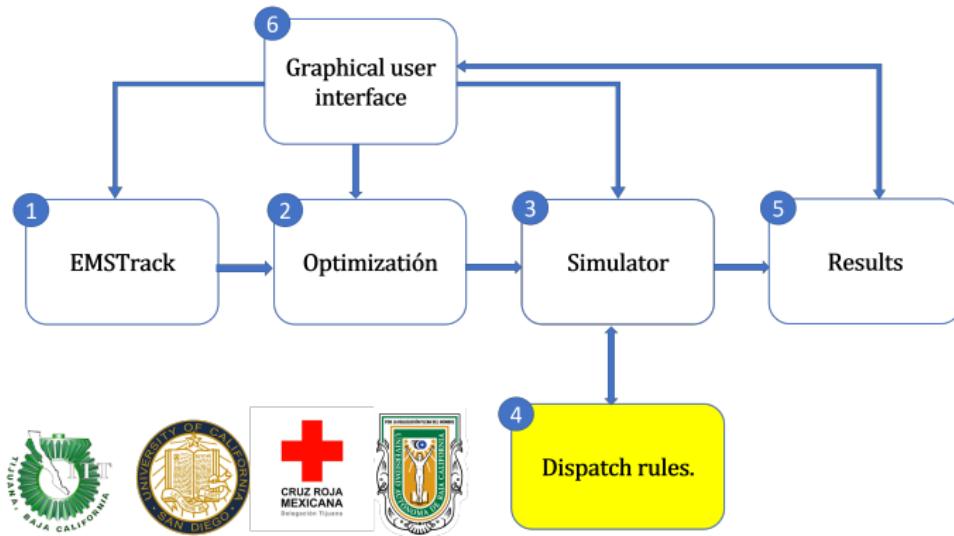


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Module 5 generates and organizes the results in a structured way for the administrative personnel in charge of decision making. And finally, Module 6 is the graphical user interface that will allow interacting with the system.

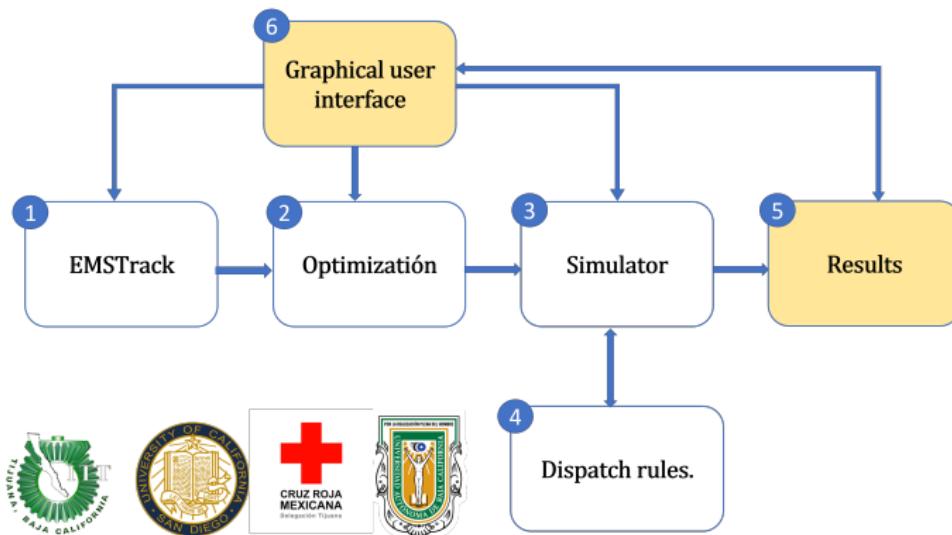


Figure 45: Tijuana Red Cross interactive System Diagram.

6.1 Updated simulator module

Updated simulator module

Returning to the simulator module in its original version.

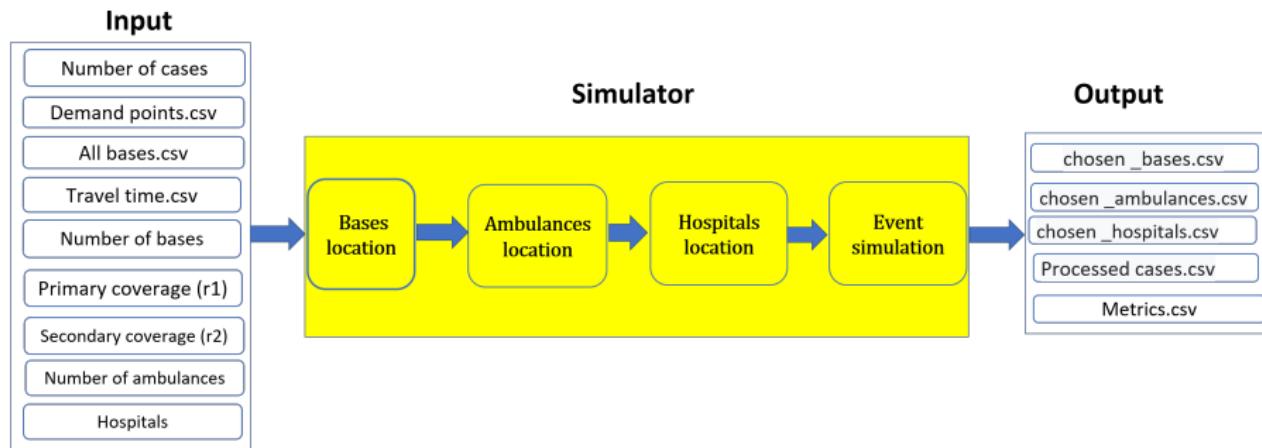


Figure 46: Simulator module.

Updated simulator module

The changes that have been made are taking into account that in the future it will be integrated into the system.

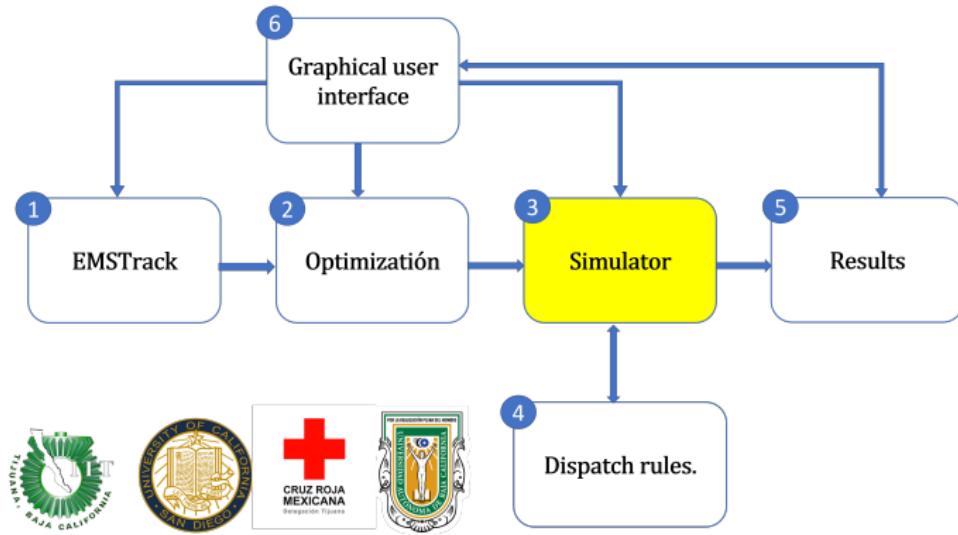


Figure 47: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The changes made are:

1. The way of establishing the number of calls or emergency cases to be simulated, which were randomly generated from time to time, was modified. Now they are loaded from a file in .csv format, which contains the calls, information provided by the Red Cross of Tijuana.

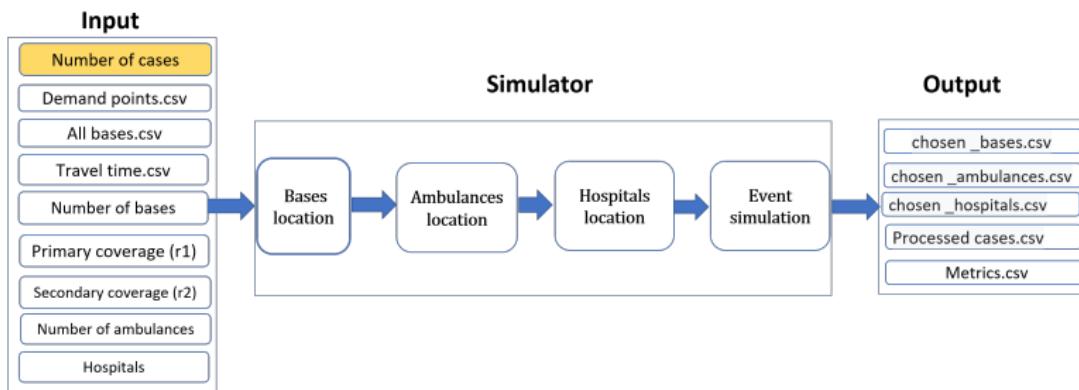


Figure 48: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The changes made are:

1. The way of establishing the number of calls or emergency cases to be simulated, which were randomly generated from time to time, was modified. Now they are loaded from a file in .csv format, which contains the calls, information provided by the Red Cross of Tijuana.

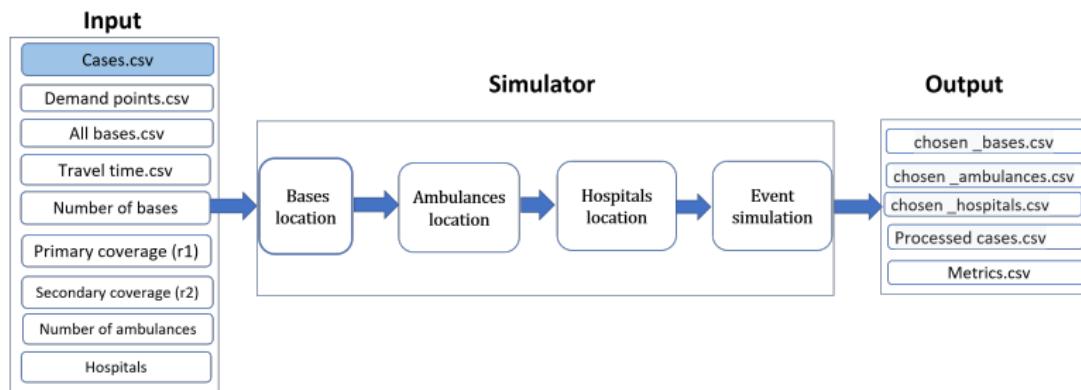


Figure 48: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The file contains:

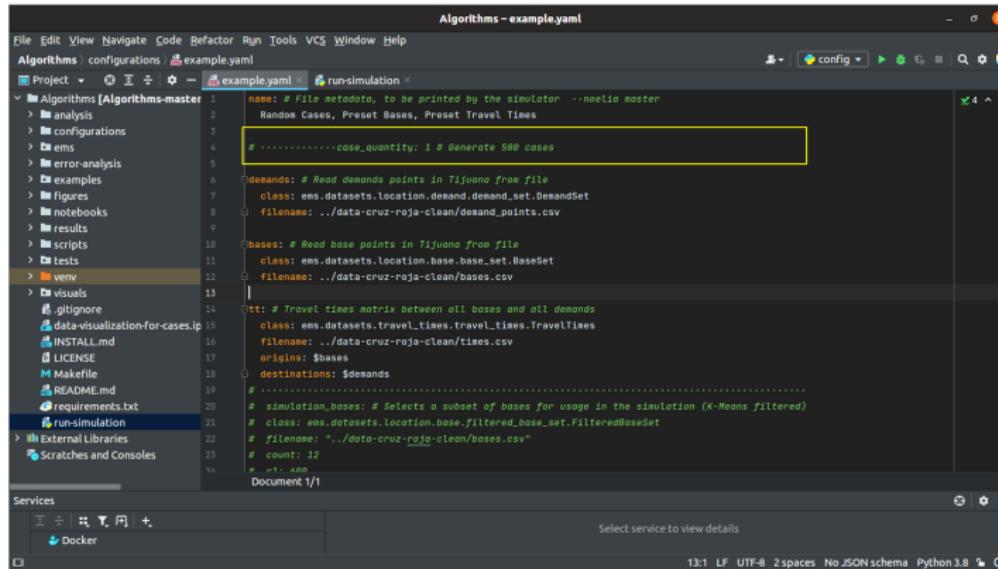
- The start time of the cases
- The coordinates of the cases
- The priority of the call, which is not yet taken into account for sending the ambulance.

A	B	C	D	E	F	G	H	I	J
1	id	date	latitude	longitude	priority				
2	1	2001-12-15 03:19:13	32.49986726216	-117.0340217613195	3				
3	2	2001-12-15 03:23:05	32.5481796866929	-116.904236150184	4				
4	3	2001-12-15 03:38:22	32.493168112529675	-116.7893637487405	4				
5	4	2001-12-15 04:16:54	32.4931681125075	-116.7893637487405	3				
6	5	2001-12-15 04:22:16	32.457342182162	-116.782261062692	4				
7	6	2001-12-15 04:30:26	32.428982706696	-116.830721874745	3				
8	7	2001-12-15 04:33:11	32.512719591939	-117.093192603295	4				
9	8	2001-12-15 04:53:24	32.5983351717913	-116.751969940876	4				
10	9	2001-12-15 05:10:41	32.4758783964451	-116.859032513876	2				
11	10	2001-12-15 05:19:05	32.62528801348	-116.89789291345	3				
12	11	2001-12-15 05:39:40	32.5301931295202	-116.90650599713	2				
13	12	2001-12-15 05:54:34	32.4931681125075	-116.7893637487405	3				
14	13	2001-12-15 05:54:34	32.390069311995	-116.997983867299	2				
15	14	2001-12-15 06:15:03	32.4219573467189	-117.00123289176	4				
16	15	2001-12-15 06:22:39	32.4926941243865	-116.789964812773	1				
17	16	2001-12-15 06:39:37	32.5386161268897	-117.050027686895	4				
18	17	2001-12-15 06:52:05	32.4895096235017	-116.763682034602	4				
19	18	2001-12-15 06:56:06	32.513703276958	-116.833367221036	3				
20	19	2001-12-15 06:59:00	32.5149391403448	-116.919251471934	4				
21	20	2001-12-15 07:11:22	32.517504948686	-116.839803990372	4				

Figure 49: Cases file.

Updated simulator module

To omit this process in the original version, we just delete the **case_quantity** that had the value of the cases to be generated and the **case** module, which generates the coordinates and the time of the case, processing now the cases from a csv file.



```

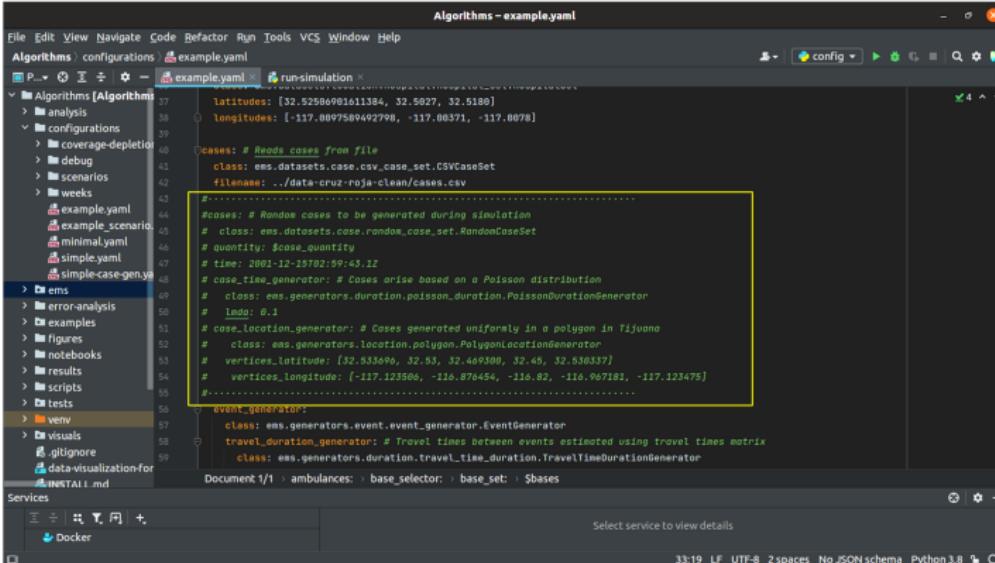
Algorithms - example.yaml
File Edit View Navigate Code Befactor Run Tools VCS Window Help
Algorithms : configurations example.yaml run-simulation
Project - example.yaml run-simulation
Algorithms [Algorithms-master]
> analysis
> configurations
> ems
> error-analysis
> examples
> figures
> notebooks
> results
> scripts
> tests
> venv
> visuals
> .gitignore
> data-visualization-for-cases.ipynb
INSTALL.md
LICENSE
Makefile
README.md
Requirements.txt
run-simulation
External Libraries
Scratches and Consoles
Services
Docker
Algorithms - example.yaml
name: # File metadata, to be printed by the simulator --noedit master
Random Cases, Preset Bases, Preset Travel Times
# ..... case_quantity: 1 # Generate 500 cases
demands: # Read demands points in Tijuana from file
  class: ems.datasets.location.demand.demand_set.DemandSet
  filename: ../data-cruz-roja-clean/demand_points.csv
bases: # Read base points in Tijuana from file
  class: ems.datasets.location.base.base_set.BaseSet
  filename: ../data-cruz-roja-clean/bases.csv
tt: # Travel times matrix between all bases and all demands
  class: ems.datasets.travel_times.travel_times.TravelTimes
  filename: ../data-cruz-roja-clean/times.csv
  origins: $bases
  destinations: $demands
# ..... simulation_bases: # Selects a subset of bases for usage in the simulation (K-Means filtered)
#   class: ems.datasets.location.base.filtered_base_set.FilteredBaseSet
#   filename: ../data-cruz_roja_clean/bases.csv"
#   count: 12
#   n: 12
Document 1/1
Select service to view details
13:1 LF UTF-8 2spaces No JSONschema Python 3.8

```

Figure 50: example.yaml source code

Updated simulator module

To omit this process in the original version, we just delete the **case_quantity** that had the value of the cases to be generated and the **case** module, which generates the coordinates and the time of the case, processing now the cases from a csv file.



```

Algorithms - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithms > configurations > example.yaml
Algorithms [Algorithms] > analysis > configurations > coverage-depletion > debug > scenarios > weeks > example.yaml > example_scenario > minimal.yaml > simple.yaml > simple-case-gen.yaml
ems > error-analysis > examples > figures > notebooks > results > scripts > tests > venv > visuals > .gitignore > data-visualization-for > INSTALL.md
Services
Algorithms > configurations > example.yaml > run-simulation > config > Docker
Algorithms - example.yaml
latitudes: [32.5296091611384, 32.5027, 32.5180]
longitudes: [-117.00897589492798, -117.00371, -117.0078]

cases: # Reads cases from file
  class: ems.datasets.case.csv.CSVCaseSet
  filename: ../data-cruz-roja-clean/cases.csv
  ...
#cases: # Random cases to be generated during simulation
# class: ems.datasets.case.random_case.set.RandomCaseSet
# quantity: $case_quantity
# time: 2003-12-15T02:09:43.1Z
# case_time_generator: # Cases arise based on a Poisson distribution
# class: ems.generators.duration.poisson.duration.PoissonDurationGenerator
# lmd: 0.1
# case_location_generator: # Cases generated uniformly in a polygon in Tijuana
# class: ems.generators.location.polygon.PolygonLocationGenerator
# vertices_latitude: [32.633496, 32.53, 32.469300, 32.45, 32.536337]
# vertices_longitude: [-117.123506, -116.876454, -116.82, -116.967181, -117.123475]
# ...
event_generator:
  class: ems.generators.event.event_generator.EventGenerator
travel_duration_generator: # Travel times between events estimated using travel times matrix
  class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
Document 1/1 - ambulances: > base_selector: > base_set: > $bases
Select service to view details
33:19 LF UTF-8 2 spaces No JSON schema Python 3.8

```

Figure 50: example.yaml source code

Updated simulator module

2. In the optimization module, there is a sub-module with a clustering algorithm, which allows to determine from a set of calls, the optimal number of clusters, called demand points, saving the results in the file demand_points.csv. This is the file currently in use.

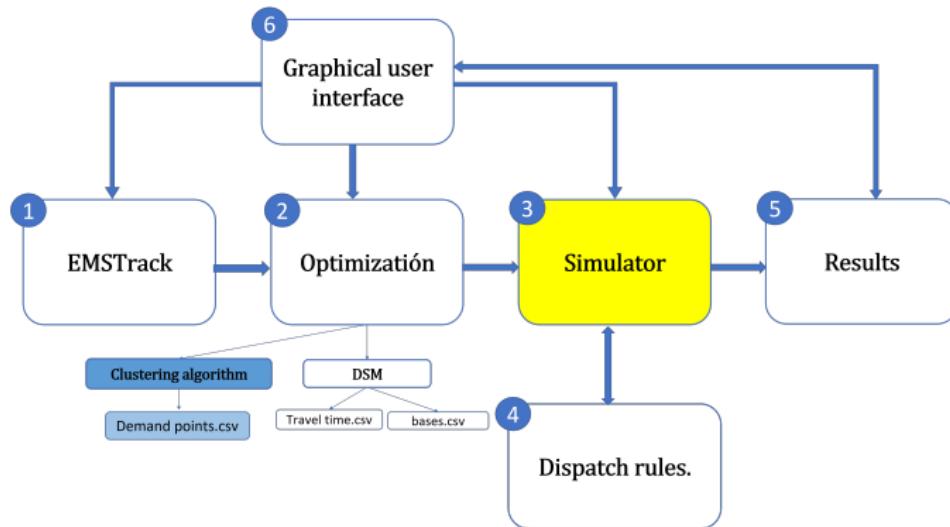


Figure 51: Tijuana Red Cross interactive System Diagram.

Updated simulator module

2. In the optimization module, there is a sub-module with a clustering algorithm, which allows to determine from a set of calls, the optimal number of clusters, called demand points, saving the results in the file demand_points.csv. **This is the file currently in use.**

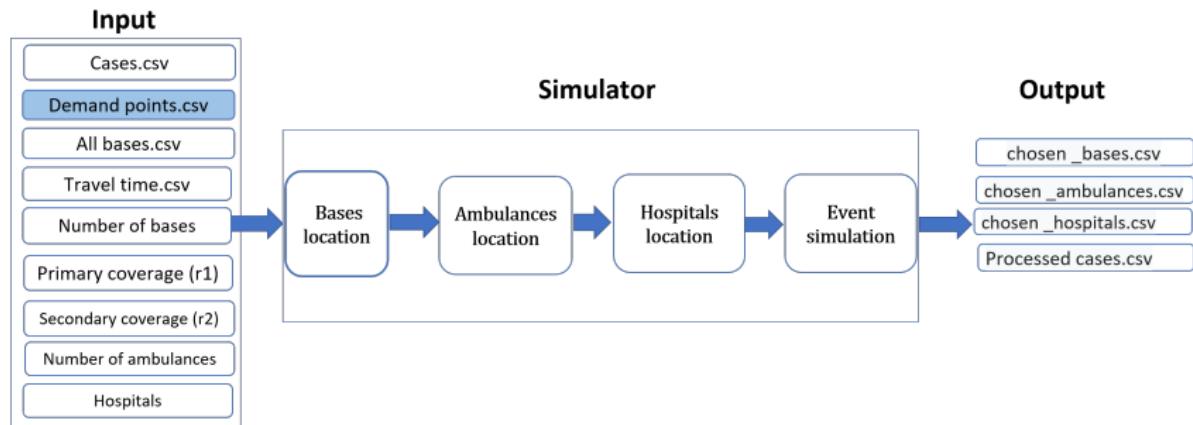


Figure 51: Simulator module.

Updated simulator module

3. Another optimization sub-module is the Double Standard Model (DSM), which maximizes the demand covered by at least two ambulances in a small radius and allows determining from a set of bases the ones that offer the best coverage, assigning a minimum of one ambulance and a maximum of two ambulances, while calculating the travel times from each of the bases to each demand point, using the Open Source Routing Machine (OSRM).

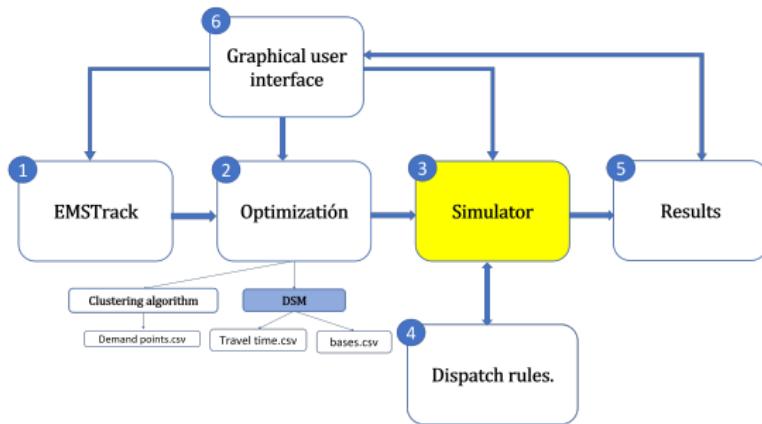


Figure 52: Tijuana Red Cross interactive System Diagram.

Updated simulator module

Saving the information in csv file.

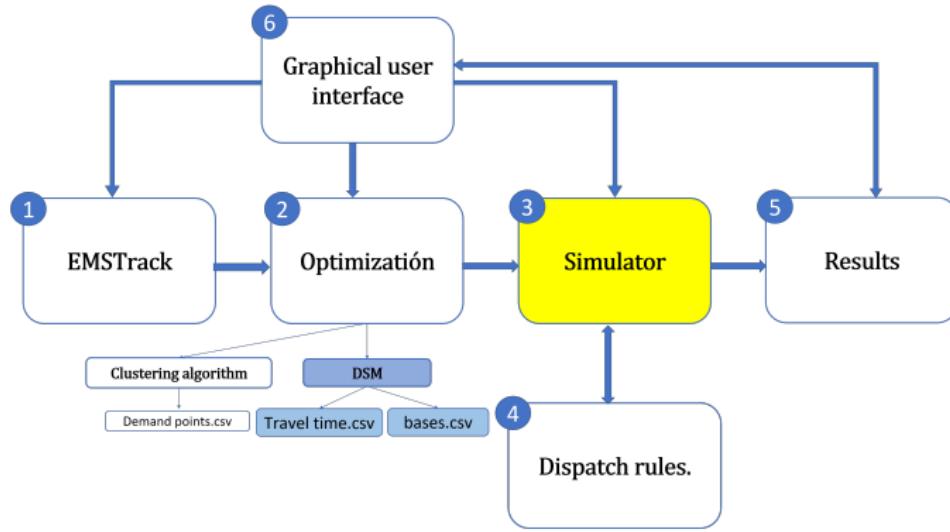


Figure 53: Tijuana Red Cross interactive System Diagram.

Updated simulator module

4. Thus, in the original version of the simulator, the block that generated the bases for dispatch was eliminated and the DSM results are used.

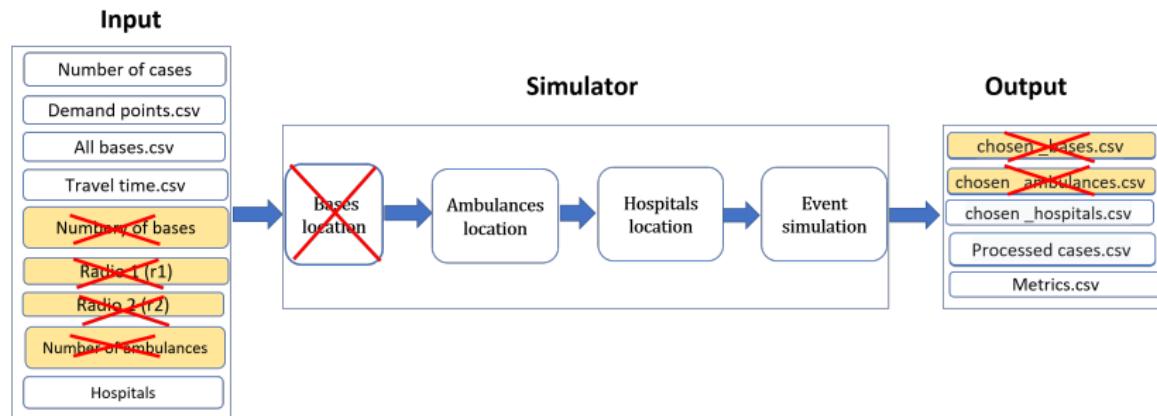


Figure 54: Simulator module.

Updated simulator module

To skip this process in the original version just remove the **simulation_bases** block.

```
Algorithms - example.yaml
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Algorithms > configurations > example.yaml
Algorithms [Algorithms]
  analysis
  configurations
    coverage-depletion
    debug
    scenarios
    weeks
      example.yaml
      example_scenario
      minimal.yaml
      simple.yaml
      simple-case-gen.yaml
    ems
    error-analysis
    examples
    figures
    notebooks
    results
    scripts
    tests
    venv
  visuals
  .gitignore
  data-visualization-for
  INSTALL.md
Services
  Docker
Algorithms - example.yaml
run-simulation x
13
14   tt: # Travel times matrix between all bases and all demands
15     class: ems.datasets.travel_times.travel_times.TravelTimes
16     filename: ./data-cruz-roja-clean/times.csv
17     origins: $bases
18     destinations: $demands
19
20   # .....
21   # simulation_bases: # Selects a subset of bases for usage in the simulation (K-Means filtered)
22   # class: ems.datasets.location.base_filtered_base_set.FilteredBaseSet
23   # filename: "../data-cruz-roja-clean/bases.csv"
24   # count: 12
25   # r1: 600
26   # r2: 840
27   # travel_times: $tt
28
29   ambulance_set: # Predefines 3 ambulances in the Tijuana area
30     class: ems.datasets.ambulance.base_selected_ambulance_set.BaseSelectedAmbulanceSet
31     count: 14
32     base_selector:
33       class: ems.algorithms.base_selectors.round_robin_selector.RoundRobinBaseSelector
34       base_set: $bases # ....$simulation_bases...
35
36   hospitals: # Predefines 3 hospitals in the Tijuana area
37     class: ems.datasets.location.hospital.hospital_set.HospitalSet
38     count: 3
39     hospital_set: $hospitals
40
41 Document 1/1 | ambulances: > base_selector: > $bases
```

Figure 55: Tijuana Red Cross interactive System Diagram.

Updated simulator module

This is how the files currently in use are obtained.

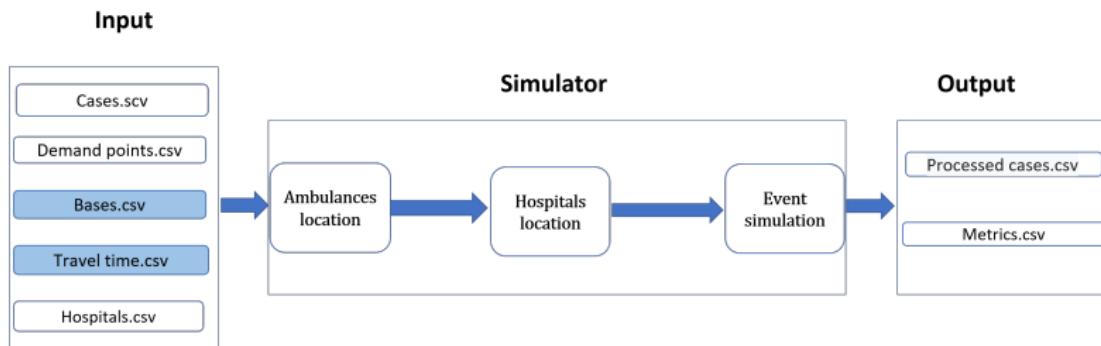


Figure 56: Simulator module.

Updated simulator module

5. Hospital locations are entered using a file with a .csv extension, because there can be many hospitals.

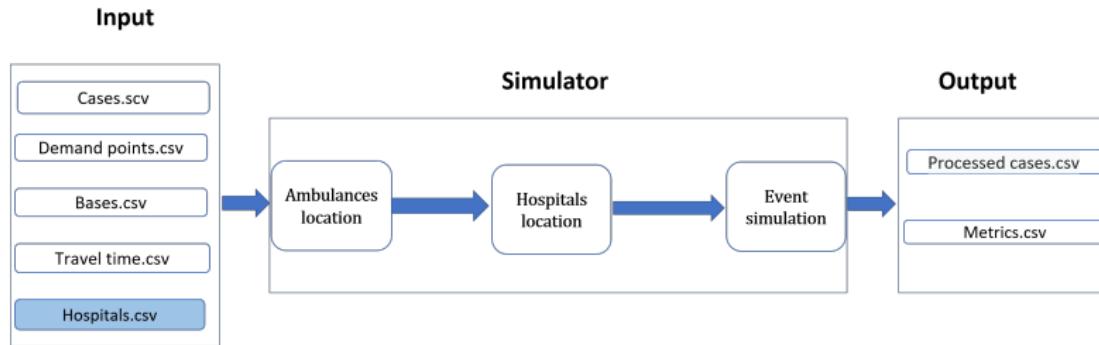
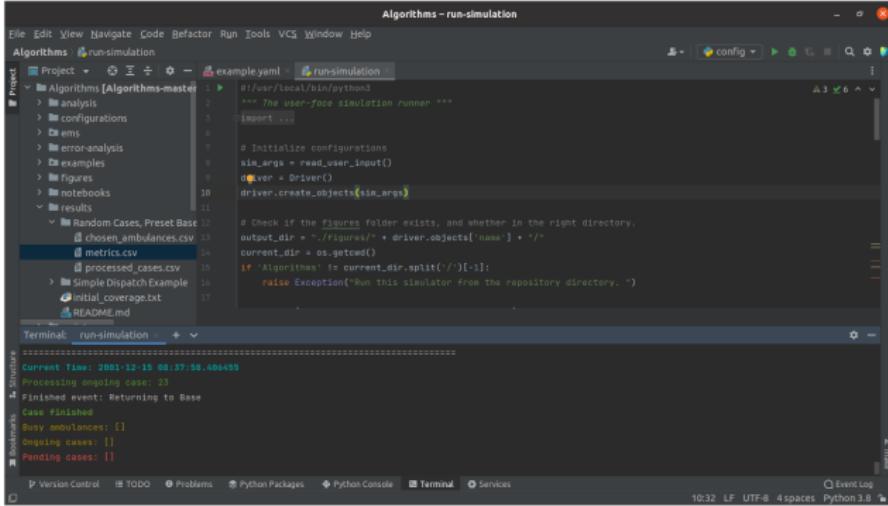


Figure 56: Simulator module.

Updated simulator module

Generating all processes and events as in the original simulator. The video shows the process of the simulator, in its current version.

[Watch video](#)



The screenshot shows a code editor window titled "Algorithms – run-simulation". The file "example.yaml" is open, containing configuration for a simulation runner. The code includes imports for `yaml` and `vcs`, initializes configurations from `sim_args`, and creates objects using `Driver.create_objects`. It also checks if the `figures` folder exists and is in the right directory. The terminal below shows the execution of the script, outputting the current time, processing details, and case status.

```
#!/usr/local/bin/python3
"""
The user-face simulation runner ...
"""

import yaml
from vcs import Driver

# Initialize configurations
sim_args = read_user_input()
driver = Driver()
driver.create_objects(sim_args)

# Check if the figures folder exists, and whether in the right directory.
output_dir = ".//figures/" + driver.objects['name'] + "/"
current_dir = os.getcwd()

if 'Algorithms' != current_dir.split('/')[-1]:
    raise Exception("Run this simulator from the repository directory. ")
```

Terminal: run-simulation

```
=====
Current Time: 2023-12-15 00:37:58.604455
Processing ongoing case: 23
Finished event: Returning to Base
Case Finished
  Busy ambulances: []
  Ongoing cases: []
  Pending cases: []
```

5. Conclusion

Conclusion

The Simulator module is functional, but lacks the basic functions required by the Tijuana Red Cross. There is already a database with information from the emergency medical services to carry out the simulations.

6. Outlook

Outlook

As future work it is proposed:

- Implement an algorithm that evolves dispatch rules, which given the current status (coverage, ambulance location, incoming call priority, etc.) determine which ambulance should be dispatched.
- For travel times, the OSRM routing engine could be implemented since it does not vary much from Google Maps according to previous experiments, in addition to having the advantage of being free.
- Change the policies for the selection of the hospital to which the ambulance must be transferred.

Thanks for your attention!!



noelia.torres@tectijuana.edu.mx

yaz.maldonado@tectijuana.edu.mx

leonardo.trujillo@tectijuana.edu.mx



TECNOLÓGICO
NACIONAL DE MÉXICO

