POLITECHNIKA WARSZAWSKA

Rozprawa doktorska

Inżynieria Lądowa, Geodezja i Transport
Dziedzina nauk Inżynieryjno-Technicznych

mgr inż.
Kamil Choromański

Ocena możliwości i efektywności prowadzenia analiz dużych zbiorów danych z użyciem cyberinfrastruktury informacji geoprzestrzennej

Promotor
dr hab. inż. **Dariusz Gotlib**, prof. uczelni

Promotor Pomocniczy
dr inż. **Bogusław Kaczałek**

Warszawa 2026

Podziękowania

Mojemu promotorowi Dariuszowi Gotlibowi, za zarażenie mnie pasją do nauki i całą przekazaną wiedzę.

Mojemu promotorowi pomocniczemu, Bogusławowi Kaczałkowi, za wsparcie merytoryczne.

Moim rodzicom, za nieustającą wiarę i wsparcie.

Oli, za cierpliwość i głos zdrowego rozsądku.

Jakubowi Łobodeckiemu, za lata wspólnych naukowych doświadczeń.

Streszczenie

Dynamiczny rozwój technologii informatycznych powoduje stały wzrost zasobów danych przestrzennych, których efektywna analiza coraz częściej wykracza poza możliwości tradycyjnych narzędzi GIS (ang. *geographic information system*). Zbiory takie określane są jako *spatial big data* (SBD). Wyzwania związane z analizą SBD realizowane są w ramach tzw. cyberinfrastruktur danych przestrzennych, integrujących zaawansowane zasoby obliczeniowe, algorytmy oraz zróżnicowane źródła danych geoprzestrzennych. Skuteczne wykorzystanie tych możliwości wymaga jednak specjalistycznej wiedzy, a w dostępnej literaturze brak jest jednoznacznych wytycznych określających, kiedy zastosowanie narzędzi SBD jest uzasadnione oraz jakie rodzaje analiz przestrzennych można efektywnie z ich użyciem wykonywać.

Celem rozprawy było przeprowadzenie badań i na ich podstawie opracowanie zaleceń umożliwiających odpowiedni dobór narzędzi analizy danych przestrzennych z uwzględnieniem wielkości zbioru, charakteru danych, rodzaju wykonywanej analizy oraz kosztu implementacji, ze szczególnym naciskiem na zbiory danych typu SBD. Przyjęta metodyka badań opiera się na eksperymentach obliczeniowych obejmujących dwa zasadnicze etapy, a całość analiz oparto o cyberinfrastrukturę danych przestrzennych CENAGIS (Platformę IT CENAGIS). W pierwszym etapie przeprowadzono porównanie tradycyjnych metod analizy przestrzennej z metodami właściwymi dla SBD. Analizę wykonano dla trzech najczęściej wykorzystywanych typów danych przestrzennych: wektorowych, rastrowych i chmur punktów, aby określić różnice w sposobie implementacji oraz w osiąganej wydajności. W drugim etapie badania skoncentrowano na analizie wydajności metod SBD oraz możliwościach ich optymalizacji. Eksperymenty przeprowadzono na zbiorach danych o mobilności pojazdów spełniających wymogi danych *big data*, weryfikując skalowalność, ograniczenia oraz potencjał optymalizacyjny badanych narzędzi. Wyniki badań posłużyły do opracowania metodyki doboru narzędzi i strategii analizy dużych zbiorów danych przestrzennych, a także do sformułowania zaleceń dotyczących dalszego rozwoju wykorzystywanej cyberinfrastruktury danych przestrzennych.

Słowa kluczowe: *spatial big data*, cyberinfrastruktura danych przestrzennych, GIS, dane o mobilności, optymalizacja doboru narzędzi

Abstract

The dynamic development of information technologies is driving a continuous increase in spatial data volumes, whose effective analysis increasingly exceeds the capabilities of traditional GIS tools. Such datasets are referred to as spatial big data (SBD). The challenges associated with SBD analysis are addressed by geospatial cyberinfrastructures, which integrate advanced computing resources, analytical algorithms and diverse spatial data sources. However, effective use of these capabilities requires specialist knowledge, and clear guidelines are lacking regarding when the use of SBD tools is justified and how the type of spatial analysis affects their cost-effectiveness. The aim of this dissertation is to develop a methodology for the optimal selection of spatial data analysis tools, taking into account dataset size and characteristics, analysis type and implementation cost, with a particular emphasis on SBD-scale datasets. The research methodology is based on computational experiments carried out in two main stages. The entire analysis is based on the CENAGIS spatial data cyberinfrastructure. In the first stage, traditional spatial analysis methods were compared with SBD-oriented approaches. The analysis covered three most commonly used types of spatial data: vector, raster and point clouds, in order to determine differences in implementation effort and performance. The second stage of the study focused on evaluating the performance of SBD methods and the possibilities for their optimisation. Experiments were conducted on very large mobility data sets, assessing the scalability, limitations and optimisation potential of the examined tools. The objective was to determine the boundaries of this technology and identify approaches enabling SBD-scale analysis at the national level. The results of the study formed the basis for developing a tool selection methodology and strategies for analysing large spatial datasets, as well as for formulating recommendations for the further development of analysed geospatial cyberinfrastructure.

Keywords: spatial big data, spatial data cyberinfrastructure, GIS, mobility data, tool selection optimization

Spis treści

Wstęp

Problem badawczy

Obecnie obserwuje się dynamiczny rozwój technologii informatycznych, w tym rozwiązań z obszaru Internetu Rzeczy (ang. *Internet of things* - IoT), sztucznej inteligencji (and. *artificial intelligence* - AI), metod zdalnej obserwacji Ziemi oraz zautomatyzowanych zdalnych pomiarów przestrzeni z użyciem np. różnorodnych kamer, radarów czy skanowania laserowego. Rozwój ten przekłada się na stały wzrost ilości gromadzonych danych, zarówno pod względem objętości, jak i różnorodności. W wielu przypadkach rozmiar i złożoność tych zbiorów przekraczają możliwości ich efektywnego przetwarzania przy użyciu klasycznych systemów informacji geograficznej (ang. *geographic information system* - GIS) działających na pojedynczych komputerowych personalnych lub stacjach roboczych.

Odpowiedzią na tę potrzebę są złożone klastry komputerowe, które można nazwać też cyberinfrastrukturami. Pojęcie cyberinfrastruktury danych geoprzestrzennych (ang. *geospatial cyberinfrastructure* - GCI) wprowadził i spopularyzował zespół Michaela Goodchilda (Yang, Raskin, Goodchild i Gahegan, 2010). Cyberinfrastruktura (CI) to połączenie zasobów danych, protokołów sieciowych, platform obliczeniowych i usług obliczeniowych, które przynoszą ludziom informacje i narzędzia obliczeniowe do prowadzenia badań oraz aplikacji opartych o duże zasoby danych w świecie napędzanym informacjami. *Geospatial* CI (GCI) odnosi się do CI, która wykorzystuje teorie i informacje geoprzestrzenne. Cyberinfrastruktury danych przestrzennych, oparte na rozproszonych zasobach obliczeniowych, wysokowydajnych systemach składowania danych oraz specjalistycznych narzędziach informatycznych, umożliwiają przetwarzanie i analizę dużych zbiorów danych przestrzennych, określanych mianem *spatial big data* (SBD). Przykładem takiej platformy w Polsce jest CENAGIS – informatyczne środowisko chmurowe integrujące otwarte technologie obliczeń rozproszonych na danych przestrzennych, którego głównym celem jest umożliwienie analiz tego typu zbiorów w dużej skali (Gotlib, Choromański i Kaczałek, 2022). W dalszej części rozprawy będzie nazywane także Platformą IT CENAGIS[1].

[1] https://cenagis.edu.pl/platforma/ (data dostępu 15.12.2025)

Technologie tego typu funkcjonują w środowisku badawczym od ponad dekady. Na świecie powstało wiele cyberinfrastruktur danych przestrzennych opartych na różnorodnych rozwiązaniach technologicznych i realizujących zróżnicowane zadania wykorzystujące duże zbiory danych przestrzennych.

Te narzędzia otworzyły nowe możliwości przeprowadzania analiz przestrzennych z użyciem zbiorów danych bardzo trudnych bądź niemożliwych do analizy z wykorzystaniem klasycznych metod, takich jak narzędzia GIS dedykowane komputerom typu desktop (np. QGIS[2] czy ESRI ArcGIS[3]) czy języki programowania takie jak Python bądź R posiadające bogatą bibliotekę pakietów pozwalających na realizację analiz przestrzennych.

Nadal jednak w literaturze nie można znaleźć wielu informacji na temat wykorzystania technologii SBD. Wieloletnie doświadczenia z działania bardzo dużego konsorcjum blisko 30 kluczowych uczelni i jednostek badawczych pn. „Sieć Naukowych Analiz Geoprzestrzennych" (SNAG) zajmujących się geoinformacją w Polsce[4] wskazują, że analizy z użyciem technologii SBD są nadal rzadkością, co wykazały m.in. badania ankietowe które zostaną opisane w dalszej części pracy oraz analiza sposobu wykorzystywania przez SNAG infrastruktury informatycznej CENAGIS. Wskazuje na to również przegląd literatury.

Z drugiej strony potrzeby w zakresie analiz danych lawinowo rosną i będą rosły w jeszcze większym zakresie. W opublikowanych badaniach naukowych z zakresu geoinformacji w Polsce trudno jednak znaleźć przykłady znaczących analiz w skali całego kraju na dużych zbiorach danych.

Dużym wyzwaniem jest optymalny dobór narzędzi do prowadzenia analiz geoprzestrzennych w zależności od specyfiki zadania, a także wielkości i złożoności zbioru danych przestrzennych. Nie istnieją jasne wytyczne dotyczące tego procesu. Wieloletnia współpraca autora rozprawy w jako Koordynatora Komputerowego Centrum Obliczeń Geoprzestrzennych CENAGIS z wieloma naukowcami w Polsce, a także kierowanie zespołami produkcyjnymi w firmie komercyjnej z branży geoinformacyjnej, prowadzi do wniosku, że nadal tylko nieliczni

[2] https://qgis.org (data dostępu 15.12.2025)
[3] https://www.arcgis.com (data dostępu 15.12.2025)
[4] https://cenagis.edu.pl/siec-naukowa-analiz-geoprzestrzennych (data dostępu 15.12.2025)

specjaliści GIS posiadają wiedzę z zakresu optymalnego wykorzystania cyberinfrastruktur danych przestrzennych. Należy mieć także na uwadze fakt, że definicja SBD jest bardzo szeroka i nie wskazuje jasno pułapu wielkości i złożoności danych, od którego podjęcie dodatkowego trudu związanego z nauką nowych technologii przyniesie realne korzyści w postaci zaoszczędzonego czasu bądź zwiększenia zakresu analizy ograniczanej dotychczas wielkością danych. W szczególności brakuje opracowań poruszających tematykę poziomu adaptacji technologii SBD przez specjalistów GIS w Polsce oraz realnego zapotrzebowania na tego typu technologie. To wszystko zdaniem autora stanowi jedną z barier ograniczających zakres badań naukowych w obszarze geoinformacji i blokuje uzyskiwanie wyników w wielu obszarach w większej skali niż dotychczas.

W związku z powyższym można postawić następujące szczegółowe pytania badawcze:

1. Jakie zbiory danych przestrzennych mogę być efektywnie przetwarzane z użyciem technologii SBD?
2. Czy dla każdego rodzaju analizy dużych zbiorów danych przestrzennych właściwe jest zastosowanie technologii SBD?
3. Czy koszt zastosowania technologii SBD jest współmierny do korzyści, a jeżeli tak, to w jakich przypadkach?

Cele pracy

Biorąc powyższe pod uwagę określono następujące cele niniejszej pracy:

1. Określenie przydatności, zapotrzebowania i zakresu wykorzystania narzędzi SBD wśród ekspertów analizujących dane przestrzenne.
2. Porównanie wydajności narzędzi SBD w odniesieniu do klasycznych metod GIS z użyciem wybranej cyberinfrastruktury danych przestrzennych i określonych typów analiz oraz sformułowanie wytycznych co do stosowania tych narzędzi.
3. Przeprowadzenie testowych analiz geoprzestrzennych bardzo dużych zbiorów danych przestrzennych (o zasięgu całej Polski) z próbą ich optymalizacji
4. Sformułowanie zaleceń metodycznych w zakresie przetwarzania dużych zbiorów danych przestrzennych.

Układ pracy

Opis prowadzonych prac badawczych i rozwojowych oraz ich wyniki przedstawione zostały dwunastu rozdziałach oraz dodatkowych załącznikach.

Rozdziały od pierwszego do piątego stanowią wprowadzenie do zagadnienia analizy dużych zbiorów danych przestrzennych, a także przegląd aktualnych prac naukowych w tym zakresie. W rozdziale pierwszym opisano definicje pojęć *big data* (BD) oraz *spatial big data* (SBD), a także przeanalizowano ich znaczenie dla rozwoju geoinformacji na przestrzeni ostatnich kilkunastu lat. Dokonano także przeglądu cech charakterystycznych tego typu danych. W rozdziale drugim omówiono narzędzia SBD, a także przedstawiono wybrane ich przykłady w celu lepszego zrozumienia aktualnie dostępnych możliwości obliczeniowych. W tym rozdziale przedstawiono także przegląd prac naukowych skupiających się na badaniu wydajności tych narzędzi. Rozdział trzeci skupia się na przeglądzie badań w ramach których ta technologia została wykorzystana do przetwarzania bardzo dużych zbiorów danych przestrzennych. W rozdziale czwartym przedstawiono relacje pomiędzy narzędziami SBD, a klasycznymi systemami GIS, uwzględniając zarówno prace porównujące wydajność obu podejść, jak i opracowania systematyzujące różnice między nimi. W rozdziale piątym opisano rolę cyberinfrastruktur danych przestrzennych jako nowoczesnych środowisk analizy przestrzennej oraz ich relacje z metodami SBD.

W rozdziale szóstym przedstawiono analizę potrzeb oraz problemów specjalistów analizy GIS w Polsce w odniesieniu do danych SBD w oparciu o przeprowadzone badania ankietowe. Rozdział siódmy zawiera zidentyfikowane w procesie przeglądu literatury oraz badań ankietowych luki badawcze stanowiące podstawę przeprowadzonych badań.

W rozdziale ósmym opisano metodykę badań ze zwróceniem uwagi na cechy wykorzystanej infrastruktury badawczej i użytych danych. Przedstawiono szczegółową metodykę dwóch głównych etapów eksperymentów: badań porównawczych GIS i SBD oraz badań wydajności środowiska SBD.

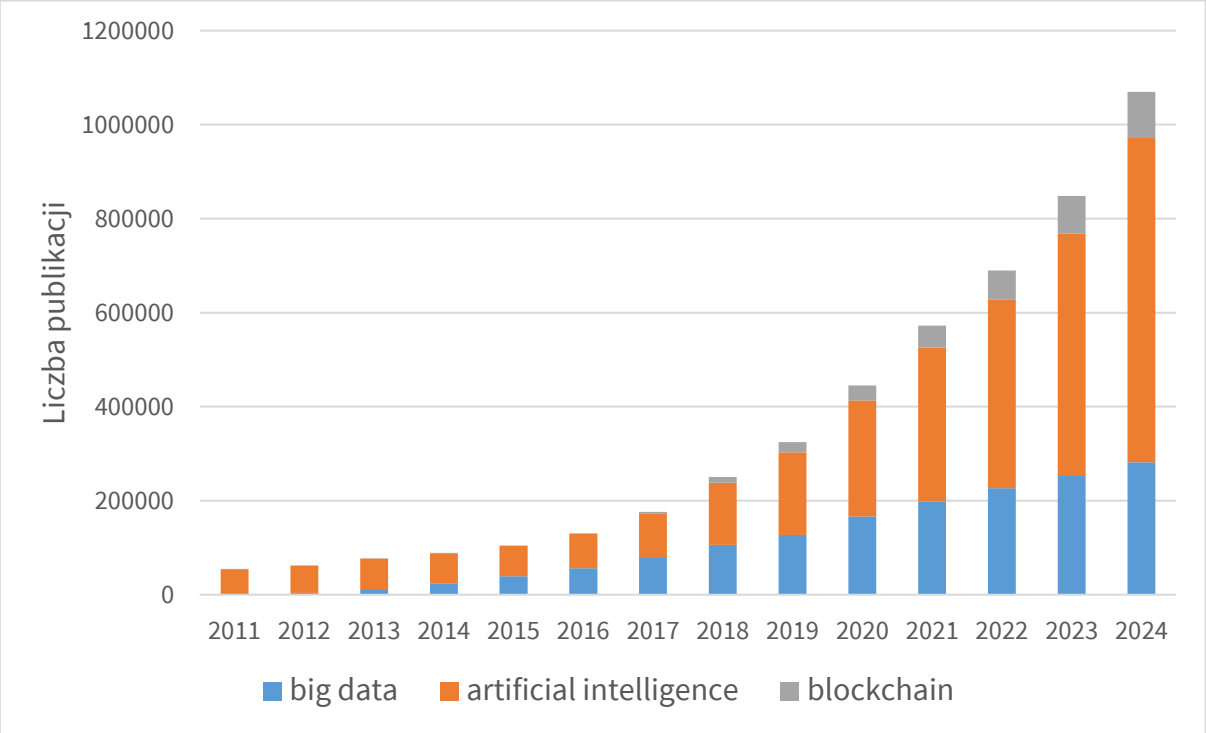
W rozdziale dziewiątym zaprezentowano wyniki badań porównawczych GIS i SBD przeprowadzonych w ramach realizacji trzech scenariuszy badawczych (A1-A3) wraz z ich interpretacją oraz podsumowaniem. W rozdziale dziesiątym przedstawiono wyniki badań

wydajności środowiska SBD, ponownie w podziale na wykonane scenariusze badawcze (B1-B5).

Rozdział jedenasty zawiera wnioski wynikające z przeprowadzonych analiz oraz praktyczne wytyczne związane z ich wynikami. W tym rozdziale przedstawiono propozycję metodyki optymalizacji doboru narzędzi w zależności od wielkości i charakteru danych, typu analiz, wymagań wydajnościowych oraz posiadanych umiejętności. Rozdział dwunasty stanowi podsumowanie całości prac.

1. Definicja spatial big data

Termin „big data" na przestrzeni ostatnich 15 lat zyskał znaczną popularność zarówno w zastosowaniach komercyjnych, jak i naukowych. Zdobył on wysoką rozpoznawalność w okolicach roku 2012 i szybko stał się tzw. *buzzword*, czyli bardzo często używanym wyrażeniem w przestrzeni medialnej, zwykle z dość rozmytym znaczeniem. Lata 2013 – 2017 przypadły na największą popularność tego terminu w mediach głównego nurtu (Pentzold i Knorr, 2024). W badaniach naukowych termin ten na szeroką skale pojawił się w zbliżonym okresie i do teraz cieszy się wysoką popularnością wśród badaczy. Na rysunku 1 przedstawiono roczną liczbę publikacji zawierających w tekście ten termin w zestawieniu z innymi technologiami, które można określić jako *buzzword* (tj. sztuczna inteligencja i *blockchain*).



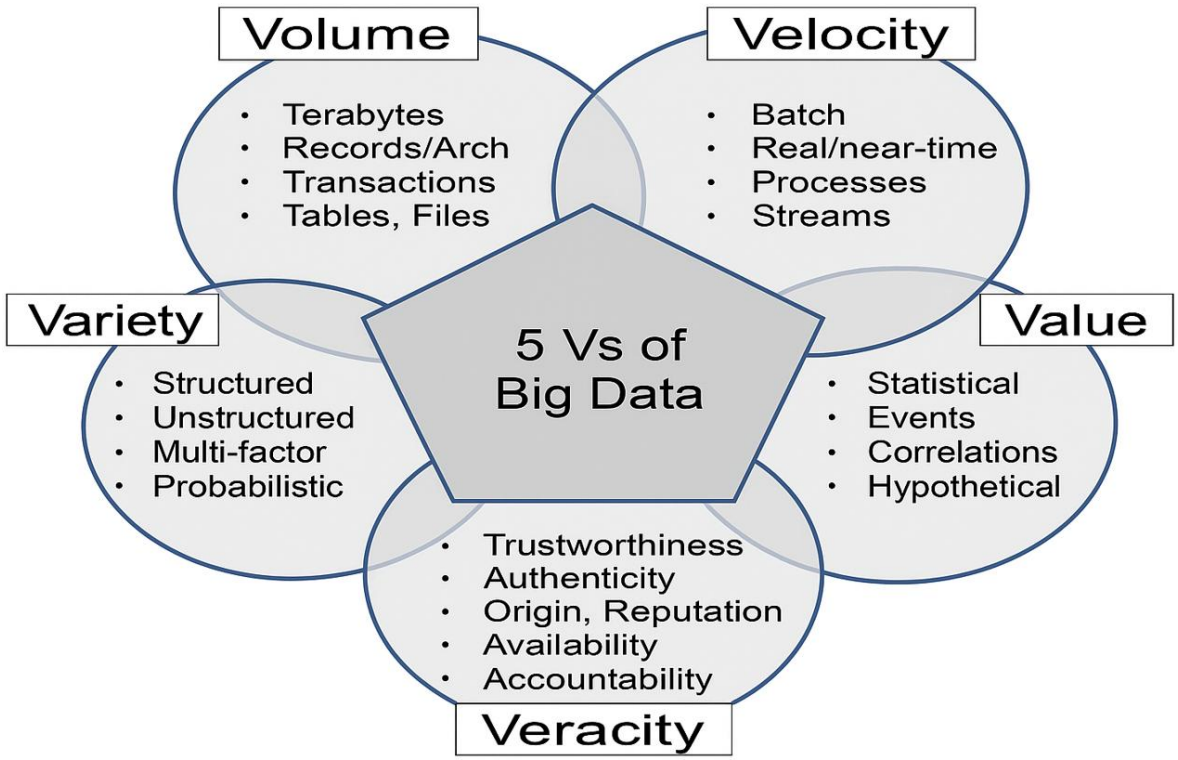
Rysunek 1. Występowanie terminów „*big data*", „*artificial intelligence*" oraz „*blockchain*" w bazie danych Dimensions[5] w latach 2011-2024 (opracowanie własne)

Zauważyć można wysoką i stale rosnącą liczbę publikacji zawierających frazę „*big data*", których liczba stanowiła w roku 2024 ponad 289 tysięcy. Jest to więc zagadnienie cieszące się wysokim

[5] Dimensions – baza danych naukowych charakteryzująca się największą liczbą agregowanych publikacji z względem innych baz tego typu. Co więcej baza ta łączy prace naukowe ze źródłami finansowania, patentami, organizacjami i naukowcami zapewniając większą przejrzystość podczas analizy źródeł naukowych (https://www.dimensions.ai/dimensions-data/, data dostępu 15.12.2025) (data dostępu 15.12.2025)

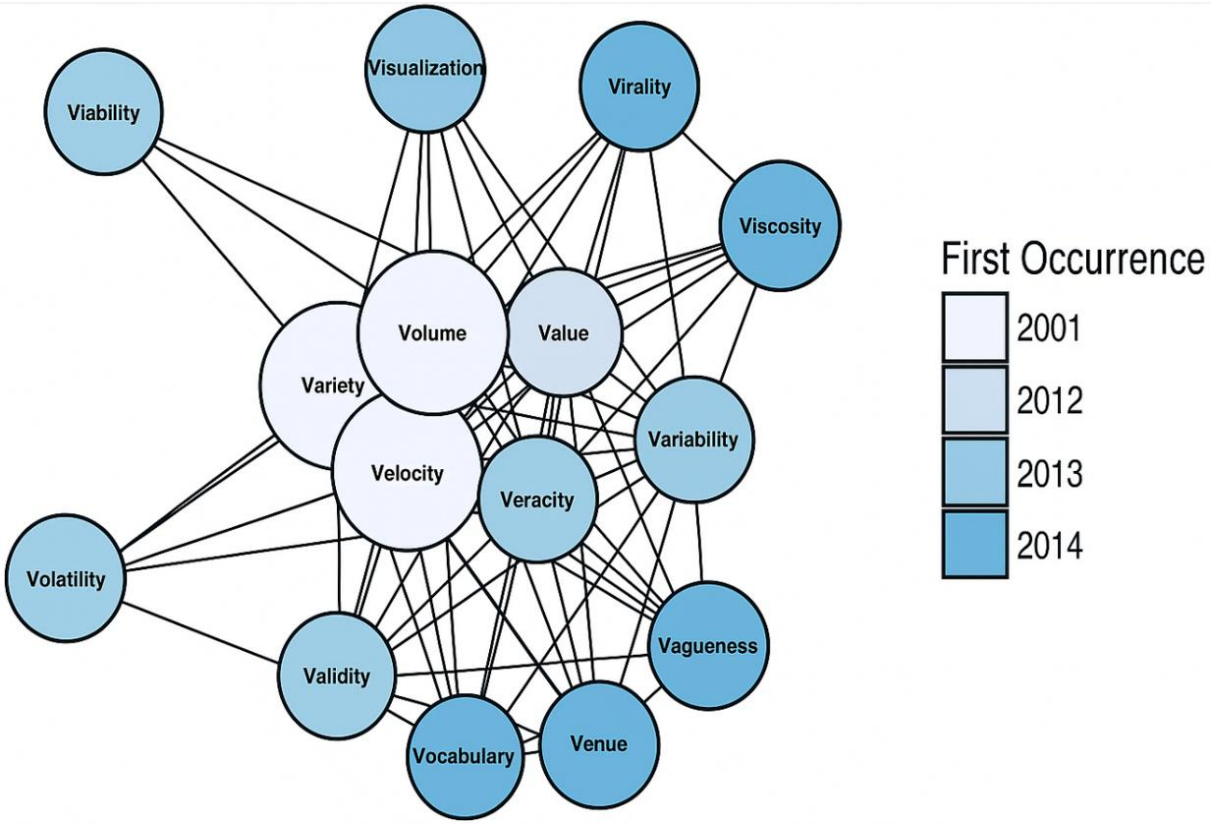
zainteresowaniem naukowym, pomimo faktu, iż termin ten nie jest już tak mocno eksploatowany w przestrzeni medialnej.

Definicja tego terminu nie jest jednak do teraz ściśle zdefiniowana i różni się w zależności od kontekstu jego zastosowania. Najczęściej zagadnienie to odnosi się do zbiorów danych tak dużych, bądź tak złożonych, że konieczne jest zastosowanie ponadstandardowych metod obliczeniowych w celu ich efektywnego przetworzenia. Pierwsze próby zdefiniowania jakie dane można uznać za big data datowane są już na 2001 rok, gdzie w badaniach (Laney, 2001) zaproponował trzy cechy charakterystyczne dużych danych: wielkość (ang. *volume*), tempo napływu danych (ang. *velocity*) i zróżnicowanie (ang. *variety*). Cechy te zyskały później miano 3V *big data*. Z biegiem czasu kolejne opracowania dodawały kolejne cechy *big data*. W literaturze odnaleźć można najczęściej definicje 5V oraz 7V. w definicji 5V do podstawowych trzech cech zaproponowano jeszcze wartość (ang. *value*) oraz wiarygodność (ang. *veracity*) (Ishwarappa i J, 2015). Pierwsze trzy cechy opisują skalę i złożoność danych, a kolejne dwie nie są bezpośrednią cechą kwalifikującą, ale raczej pokazują wyzwania analityczne. Na rysunku 2 przedstawiono dodatkowo szczegółowe hasła techniczne związane z 5V. W przypadku definicji 7V do powyższych cech dodano także zmienność (ang. *volatility*) oraz ważność (ang. *validity*).



Rysunek 2. Opis cech *big data* zgodnie z definicją 5V (Ishwarappa i Anuradha, 2015)

W literaturze odnotowywane były dalsze rozszerzenia liczby „V" definiujących duże zbiory danych, mające na celu lepsze scharakteryzowanie czym jest *big data* i kiedy należy myśleć o stosowaniu narzędzi dedykowanych tego typu danym. W artykule "The 42 V's of Big Data and Data Science" autor dokonał analizy publikacji podejmujących się prób definicji kolejnych cech charakterystycznych doliczając się nawet kilkunastu cech zaproponowanych przez różnych autorów (Shafer, 2017). Podsumowanie odnalezionych przez autora cech zaprezentowano na rysunku 3.



Rysunek 3. Podsumowanie cech charakterystycznych big data występujących w opracowaniach naukowych na przestrzeni lat 2001-2014 (Shafer, 2017)

Autor powyższego artykułu ironicznie definiuje też „42 V" cechujące duże dane, podkreślając tym samym istotną wadę prób definiowania tego zagadnienia w oparciu o rozbudowane zestawy cech. Liczbę ogólnych cech charakterystycznych *big data* można rozszerzać, nie zawężając znaczenia samego terminu, ani nie ułatwiając doboru właściwych narzędzi do posiadanych danych. Problem ten zauważono również w (Emmanuel i Stanier, 2016). Metaanaliza 34 publikacji (Han, Gstrein i Andrikopoulos, 2024) wskazuje, że takie definiowanie *big data* prowadzi do problemów w rozumieniu tego terminu przez reprezentantów różnych dziedzin utrudniając sprawną komunikację pomiędzy specjalistami pracującymi z danymi.

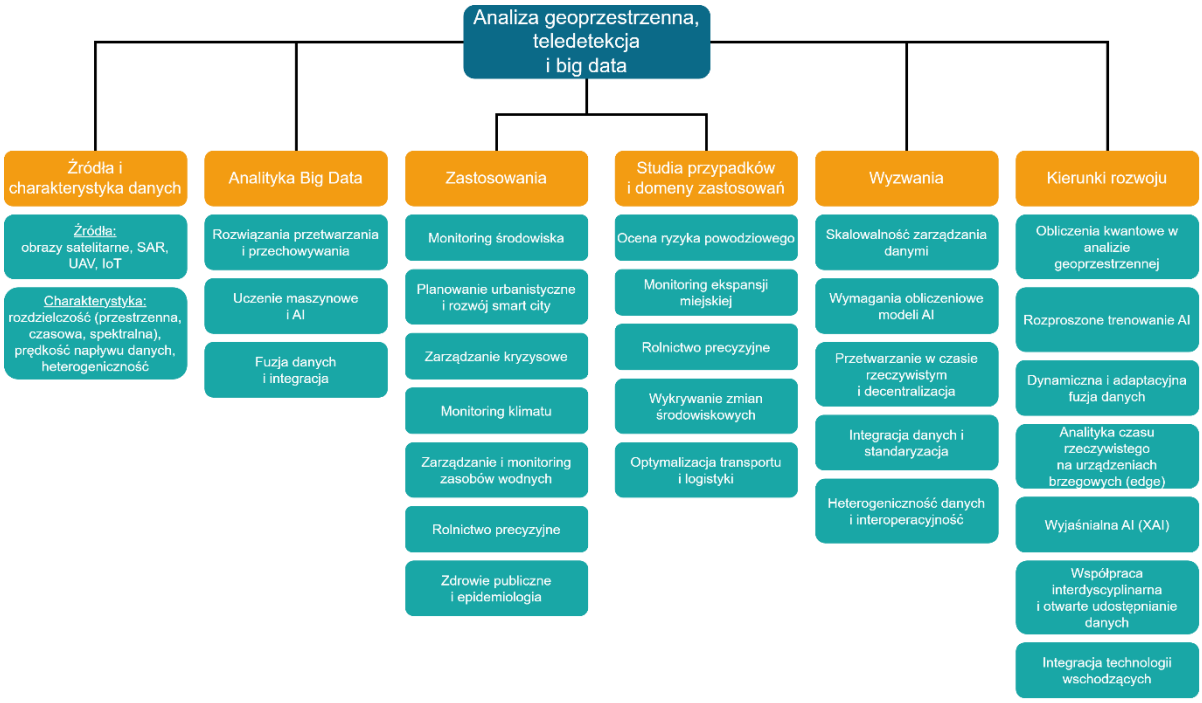
Opracowanie to wskazuje na następujące główne spotykane interpretacje tego pojęcia:

- *big data* rozumiane wyłącznie w kategoriach wolumenu informacji, bez bezpośredniego odniesienia do stosowanych technologii przetwarzania
- *big data* jako algorytmy analityczne odnoszące się do metod statystycznych, uczenia maszynowego czy uczenia głębokiego (np. lasy losowe, sieci neuronowe)
- *big data* jako ekosystem technologii pozwalający na rozproszone, skalowalne przetwarzanie danych

W części publikacji pojęcie to wykorzystywane jest również bez powiązania z konkretnymi rozwiązaniami technologicznymi, pełniąc przede wszystkim funkcję marketingową.

Wysoka popularność medialna tego określenia w latach 2013 – 2017, a także bardzo szeroki zakres znaczeniowy spowodował, że określenie wolumenu bądź złożoności danych od których można mówić o *big data* jest bardzo trudne i zależy od konkretnej dziedziny.

Identyczne problemy, związane z samą definicją pojawiają się w przypadku dużych danych przestrzennych – *spatial big data* . Również i w tym przypadku definicja jest bardzo szeroka i łączy w sobie zagadnienie przechowywania i analizy dużych zbiorów danych o charakterze przestrzennym (np. obrazy satelitarne, chmury punktów, dane wektorowe czy modele 3D), algorytmy analityczne umożliwiające wydajne przetwarzanie tych zbiorów (np. indeksowanie przestrzenne) oraz technologie pozwalające na przetwarzanie danych przestrzennych w sposób rozproszony z wykorzystaniem wielu maszyn obliczeniowych na raz np. GeoMesa (Hughes, Annex, Eichelberger, Fox i Ronquest, 2015), GeoTrellis (Kini i Emanuele, 2014), Apache Sedona (Yu, Wu i Sarwat, 2015). W pracy (Dritsas i Trigka, 2025) dokonano analizy kluczowych źródeł danych, aplikacji, przypadków użycia, wyzwań i przyszłych kierunków rozwoju technologii SBD (rysunek 4).

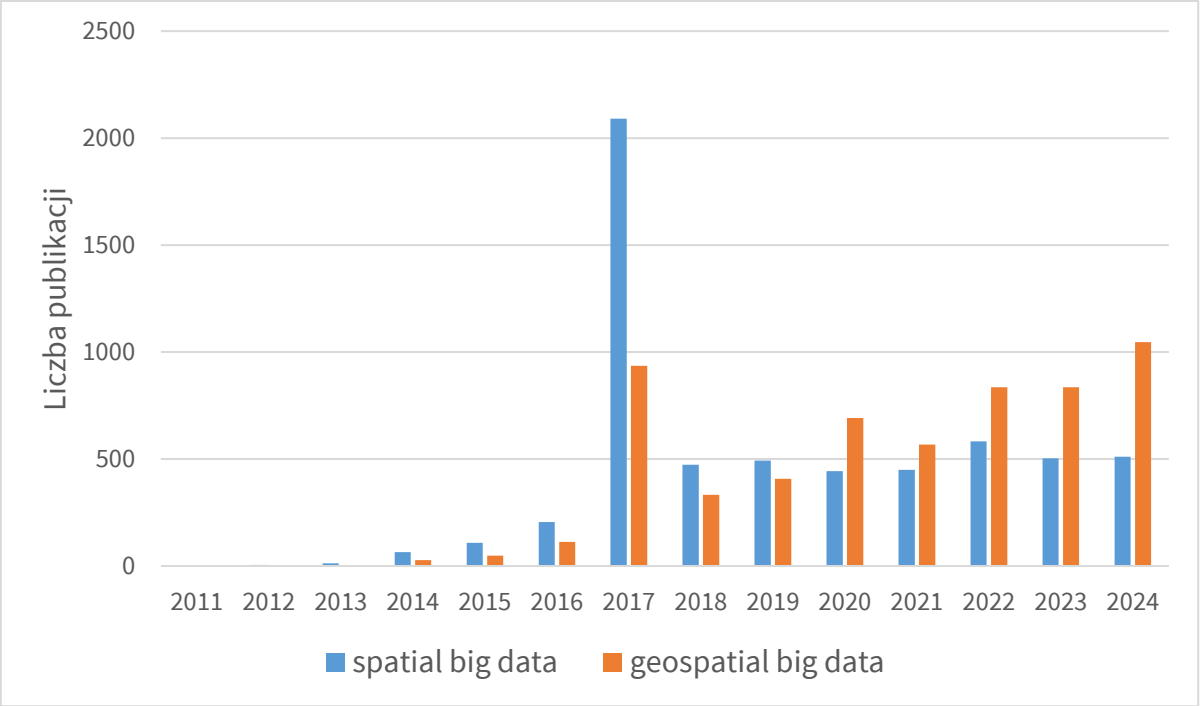


Rysunek 4. Zestawienie źródeł danych, aplikacji, przypadków użycia, wyzwań i przyszłych kierunków rozwoju powiązanych z technologiami SBD na podstawie (Dritsas i Trigka, 2025)

W tym samym artykule autorzy zauważają, że obok zagadnień dotyczących sprawnego przechowywania, integracji i przetwarzania danych zauważyć można też silne zwrócenie uwagi na integracje SBD z narzędziami oraz algorytmami szeroko rozumianej sztucznej inteligencji.

Podobnie jak w przypadku definicji *big data,* także definicja SBD nie dostarcza precyzyjnych wytycznych dotyczących pułapu danych i typów analiz wymagających zastosowania tych technologii.

Terminy związane z analizą dużych danych przestrzennych cieszą się również dużą popularnością wśród naukowców (rysunek 5), jest ona jednak znacznie mniejsza niż bardziej ogólnego terminu *big data*. Należy jednak pamiętać, że terminy te są bardzo szerokie i mają różnorodne definicje w zależności od konkretnej pracy naukowej.



Rysunek 5. Występowanie terminu *'spatial big data'* oraz *'geospatial big data'* w tekście prac naukowych w bazie danych Dimensions w latach 2011-2024 (opracowanie własne)

Zauważyć należy wyraźny skok popularności terminów związanych z SBD w roku 2017, na który przypadał również szczyt popularności medialnej tego zagadnienia. Następnie, po spadku w latach kolejnych, liczba publikacji dotyczących SBD rośnie regularnie. W ostatnich latach większą popularnością w opracowaniach naukowych cieszy się bardziej specyficzny termin „*geospatial big data*". Ze względu na nadal większą powszechność użycia sformułowania „*spatial big data*" przyjęto je do wykorzystania w niniejszej pracy.

Analiza prac przeglądowych, metaanaliz i definicji terminów *big data* oraz *spatial big data* pozwala na ogólne zrozumienie jakie typy i wielkości danych mogą wymagać zmiany podejścia do przetwarzania danych. Próba doprecyzowania tych wartości granicznych jest jednym z istotnych celów niniejszej pracy.

2. Narzędzia przetwarzania SBD

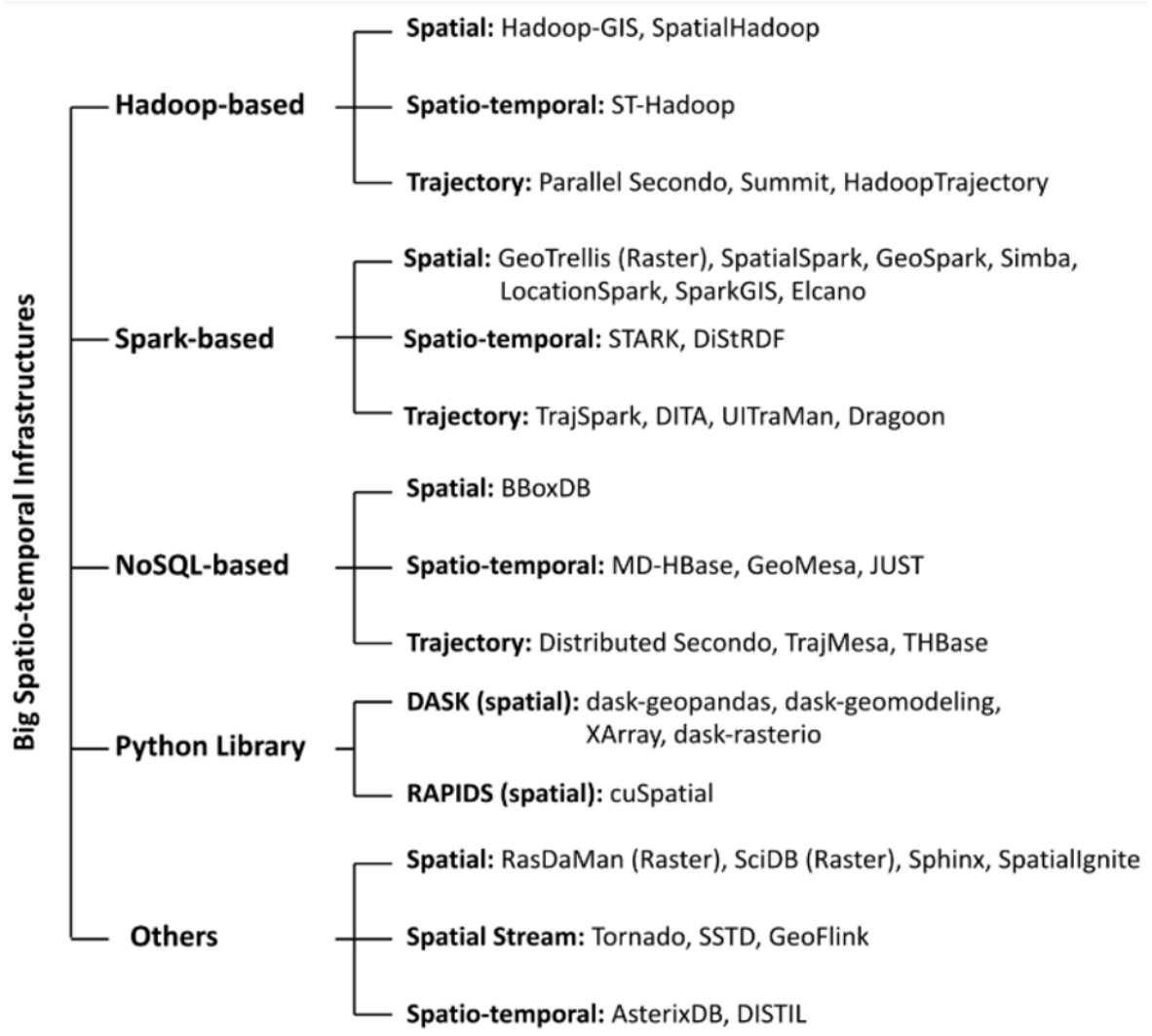
Rozwój analiz przestrzennych z wykorzystaniem dużych zbiorów danych doprowadził do opracowania wielu technologii zdolnych do ich przetwarzania. Narzędzia te rozwijały się w licznych dziedzinach i z myślą o osiągnięciu różnych celów. Duża część technologii jest oparta o oprogramowanie o otwartym kodzie źródłowym (ang. *open source*) np. technologie oparte o Apache Spark (Matei, Mosharaf, Michael, Scott i Ion, 2010) czy Apache Hadoop (Shvachko, Kuang, Radia i Chansler, 2010), ale dużą popularnością cieszą się też rozwiązania komercyjne, takie jak Google Earth Engine (Gorelick i inni, 2017). W tym rozdziale przedstawiono klasyfikację i przykłady narzędzi SBD, a także badania dotyczące ich porównań.

2.1. Kategorie narzędzi

Mówiąc o technologiach *big data* typu *open source* autorzy różnych opracowań mają na myśli odmienne kategorie narzędzi informatycznych. Jedną z prób ich sklasyfikowania w kontekście technicznym przedstawiono w (Alam, Torgo i Bifet, 2022):

- Rozwiązania oparte o technologię Hadoop
- Rozwiązania oparte o technologię Spark
- Bazy danych NoSQL
- Biblioteki języka Python
- Inne, specjalizowane rozwiązania

Na rysunku 6 przedstawiono przykłady konkretnych rozwiązań technologicznych reprezentujących każdą z kategorii.



Rysunek 6. Zestawienie kategorii narzędzi SBD z wyszczególnieniem przykładów technologii (Alam, Torgo i Bifet, 2022)

Apache Hadoop został zaprezentowany w roku 2006 i stanowił pierwszą kompletną i szeroko dostępną implementacje algorytmu MapReduce (Dean i Ghemawat, 2004) pozwalającego na znaczne ułatwienie wykonywania obliczeń rozbitych na wiele maszyn obliczeniowych na raz.

Pojawienie się oprogramowania Apache Spark w 2010 roku pozwoliło na dalszą optymalizację obliczeń rozproszonych przy zachowaniu bardzo zbliżonych głównych założeń. Swoją popularność Spark zawdzięcza znacznemu (nawet stukrotnemu) zwiększeniu wydajności w porównaniu do środowiska Hadoop dzięki optymalizacji wykorzystania pamięci RAM. Naturalnie pojawiły się więc pakiety oprogramowania wspierające analizy przestrzenne oparte o to środowisko. Do głównych reprezentantów tej grupy zaliczyć można Apache Sedona (wcześniej GeoSpark) czy GeoMesa do analiz danych wektorowych oraz GeoTrellis do analiz danych rastrowych.

Równolegle rozwijane były również bazy danych typu NoSQL (zwane nierelacyjnymi) oraz ich rozszerzenia ułatwiające obsługę bardzo dużych zbiorów danych przestrzennych. W tej grupie również można wymienić pakiet GeoMesa, gdyż obsługuje on zarówno analitykę opartą o Spark, jak i różnorodne bazy danych NoSQL takie jak Accumulo (Sen, Farris i Guerra, 2014) czy Cassandra (Lakshman i Malik, 2010).

Nie można także pominąć bibliotek dla języka Python, które pozwalają na optymalizację obliczeń przestrzennych na wielu wątkach jednego komputera, ale także na wielu komputerach na raz. Najpopularniejszym pakietem oprogramowania realizującym to zadanie jest Dask (Rocklin, 2015) oraz jego rozszerzenia przestrzenne takie jak dask-geopandas[6]. Podejście to różni się znacznie od filozofii wprowadzonej przez Apache Hadoop i rozwijanej w ramach Apache Spark. W tym przypadku głównym celem biblioteki było rozbicie obliczeń na wiele wątków procesora pojedynczej maszyny w miejsce obliczeń jednowątkowych. Następnie to podejście rozszerzono na wiele maszyn.

W tej grupie narzędzi znajdują się również pakiety oprogramowania optymalizujące obliczenia z wykorzystaniem jednostek GPU takie jak moduł cuSpatial z pakietu RAPIDS (NVIDIA, 2023).

Należy zauważyć, że wraz z rozwojem tych technologii zbliżyły się one do siebie pod względem oferowanych funkcji, a wiele z nich jest ze sobą kompatybilnych. Przykładowo pakiet GeoMesa może być wykorzystany w obliczeniach Apache Spark, przy współpracy z bazami danych NoSQL (dokumentowymi bądź plikowymi), a Apache Spark pozwalają na wykonanie kodu języka Python na wielu maszynach w sposób zbliżony do tego, jak jest to realizowane w Dask.

2.2. Analiza wybranych narzędzi SBD

W celu lepszego zrozumienia działania, możliwości oraz oceny dostępnych narzędzi SBD przeprowadzono analizę najważniejszych pakietów oprogramowania z tego obszaru, ze szczególnym uwzględnieniem rozwiązań opartych na platformach Hadoop i Spark. Zestawienie przeanalizowanych rozwiązań zaprezentowano w tabeli 1

.

[6] https://dask-geopandas.readthedocs.io (data dostępu 15.12.2025)

Tabela 1. Zestawienie narzędzi SBD z uwzględnieniem roku wydania, obecnego stanu rozwoju, wykorzystywanego języka/języków programowania, rozwiązań bazowych oraz rodzajów obsługiwanych danych (opracowanie własne)

Narzędzia Spatial Big Data										
Nazwa	Hadoop-GIS	Spatial Hadoop	GeoSpark (Sedona)	GeoMesa Spark	Magellan	IQmulus	GeoTrellis	Simba	Location Spark	Raster Frames
Rok wydania	2013	2014	2015	2015	2015	2015	2016	2016	2016	2017
Status	Nieutrzymywany (ostatnia aktualizacja w 2013)	Nieutrzymywany (ostatnia aktualizacja w 2021)	Stale rozwijany	Stale rozwijany	Nieutrzymywany (ostatnia aktualizacja w 2021)	Nieutrzymywany (ostatnia aktualizacja w 2017)	Stale rozwijany	Nieutrzymywany (ostatnia aktualizacja w 2018)	Nieutrzymywany (ostatnia aktualizacja w 2017)	Niejasny (ostania aktualizacja w 2023)
Język programowania	Java	Java	Scala/ Java/ Python/R	Scala/ Java/ Python	Scala	Scala	Scala	Scala	Scala	Python
Rozwiązanie bazowe	Apache Hadoop	Apache Hadoop	Apache Spark	Apache Spark	Apache Spark	Apache Spark	Apache Spark	Apache Spark	Apache Spark	Apache Spark/ GeoMesa/ GeoTrellis
Rodzaj danych	Dane wektorowe	Dane wektorowe	Dane wektorowe	Dane wektorowe	Dane wektorowe	Chmury punktów	Dane rastrowe	Dane wektorowe	Dane wektorowe	Dane rastrowe oraz wektorowe

Na przestrzeni lat powstało wiele narzędzi wspierających analizy przestrzenne w środowisku rozproszonym. Największy rozwój tego typu narzędzi przypada na lata 2015 – 2018. W latach późniejszych wiele inicjatyw *open source* przestało być rozwijanych w naturalnym procesie konkurowania zbliżonych rozwiązań. Takie narzędzia to Hadoop-GIS (Aji i inni, 2013), SpatialHadoop (Eldawy i Mokbel, 2015), Magellan[7], Simba (Xie i inni, 2016), LocationSpark (Tang i inni, 2020) i IQmulus (Olasz, Thai Nguyen, Giachetta i Kristóf, 2016).

Obecnie, wszystkie aktywnie rozwijane narzędzia opierają się o środowisko Apache Spark. Do dwóch głównych zbiorów oprogramowania zaliczyć można Apache Sedona oraz narzędzia rozwijane przez społeczność LocationTech[8] (GeoMesa, GeoTrellis). Zdecydowana większość badanych rozwiązań skupia się na analizie danych wektorowych. W przypadku danych rastrowych, jedynym stale rozwijanym oprogramowaniem jest pakiet GeoTrellis. Nie istnieje również żadne szeroko rozpowszechnione oprogramowanie dedykowane analizie SBD chmur punktów. Od kilku lat nie obserwuje się już też powstawania nowych narzędzi w tej grupie.

Kolejną grupą narzędzi *spatial big data* są systemy komercyjne takie jak Google Earth Engine czy ArcGIS GeoAnalytics Engine[9]. Na przykładzie tych dwóch narzędzi zauważyć można dwa różne podejścia do analiz przestrzennych w dużej skali. W tabeli 2 przedstawiono charakterystykę obu rozwiązań.

W przypadku Google Earth Engine (GEE) udostępniana jest cała infrastruktura obliczeniowa oferująca dostęp do danych przestrzennych z całego świata oraz własne narzędzia programistyczne. Jest to klasyczna usługa chmurowa typu SaaS (ang. *Software as a Service*), w której użytkownik definiuje jedynie logikę analizy, natomiast przechowywanie danych, skalowanie obliczeń oraz zarządzanie zasobami realizowane są po stronie dostawcy. Platforma ta cieszy się bardzo dużą popularnością wśród badaczy ze względu na duży stopień integracji danych oraz algorytmów, brak konieczności posiadania własnej infrastruktury obliczeniowej czy stosunkowo niski próg wejścia w zakresie umiejętności technicznych użytkownika (Tamiminia i inni, 2020). W GEE wykorzystywane i analizowane są w głównej mierze dane o charakterze rastrowym, w szczególności dane satelitarne. Dostęp do platformy możliwy jest

[7] https://github.com/harsha2010/magellan (data dostępu 15.12.2025)
[8] https://projects.eclipse.org/projects/locationtech (data dostępu 15.12.2025)
[9] https://developers.arcgis.com/geoanalytics (data dostępu 15.12.2025)

poprzez środowisko JavaScript w przeglądarce oraz interfejs Python, co pozwala na budowanie bardziej złożonych analiz.

ArcGIS GeoAnalytics Engine stanowi przykład rozwoju i transfomacji klasycznych narzędzi GIS do środowiska przetwarzania rozproszonego. Rozwiązanie łączy mechanizmy analizy przestrzennej dostępne w ekosystemie ArcGIS z architekturą Apache Spark, umożliwiając wykonywanie obliczeń na bardzo dużych zbiorach danych bez konieczności przenoszenia ich do pojedynczego stanowiska roboczego. Narzędzie udostępnia operatory przestrzenne w postaci funkcji działających na strukturach Spark, co pozwala na wykorzystanie planowania zapytań i skalowania właściwego dla Spark. Zakres dostępnych narzędzi obejmuje m.in. złączenia przestrzenne, agregacje w siatkach, analizy czasowo-przestrzenne oraz sieciowe. Wspiera on przetwarzanie danych wektorowych pochodzących zarówno z usług ArcGIS, jak i z otwartych formatów *big data*. Jednocześnie rozwiązanie pozostaje silnie powiązane z ekosystemem ESRI i z architekturą Spark, co w naturalny sposób wiąże elastyczność z wymaganiami licencyjnymi oraz środowiskowymi.

Tabela 2. Zestawienie wybranych komercyjnych narzędzi SBD z uwzględnieniem roku wydania, obecnego stanu rozwoju, wykorzystywanego języka/języków programowania, rozwiązań bazowych oraz rodzajów obsługiwanych danych (opracowanie własne)

Nazwa	Google Earth Engine	ArcGIS GeoAnalytics Engine
Rok wydania	2010	2022
Stan obecny	Stale rozwijany	Stale rozwijany
Język programowania	JavaScript, Python	Python
Rozwiązanie Bazowe	Earth Engine	Apache Spark
Rodzaj danych	Dane rastrowe	Dane wektorowe

Należy jednak zauważyć, że przedstawione powyżej narzędzia to tylko podzbiór wszystkich dostępnych narzędzi realizujących zadania rozproszonego przetwarzania danych

przestrzennych. Przykładem może być pakiet dask-geopandas integrujący możliwości klasycznych analiz w bibliotece geopadnas z system obliczeń rozproszonych Dask.

2.3. Wydajność narzędzi SBD

Na przestrzeni lat opublikowano wiele prac mających na celu porównanie wydajności poszczególnych narzędzi SBD. W celu lepszego zrozumienia ograniczeń i możliwości poszczególnych narzędzi przeanalizowano 12 tego typu publikacji z lat 2016 - 2023, gdzie porównano ze sobą przynajmniej dwie technologie SBD. W tabeli 3 przedstawiono zestawienie porównywanych technologii, zastosowanych zasobów obliczeniowych oraz analizowanych danych w ramach badań porównawczych.

Tabela 3. Zestawienie metadanych badań porównawczych technologii SBD (opracowanie własne)

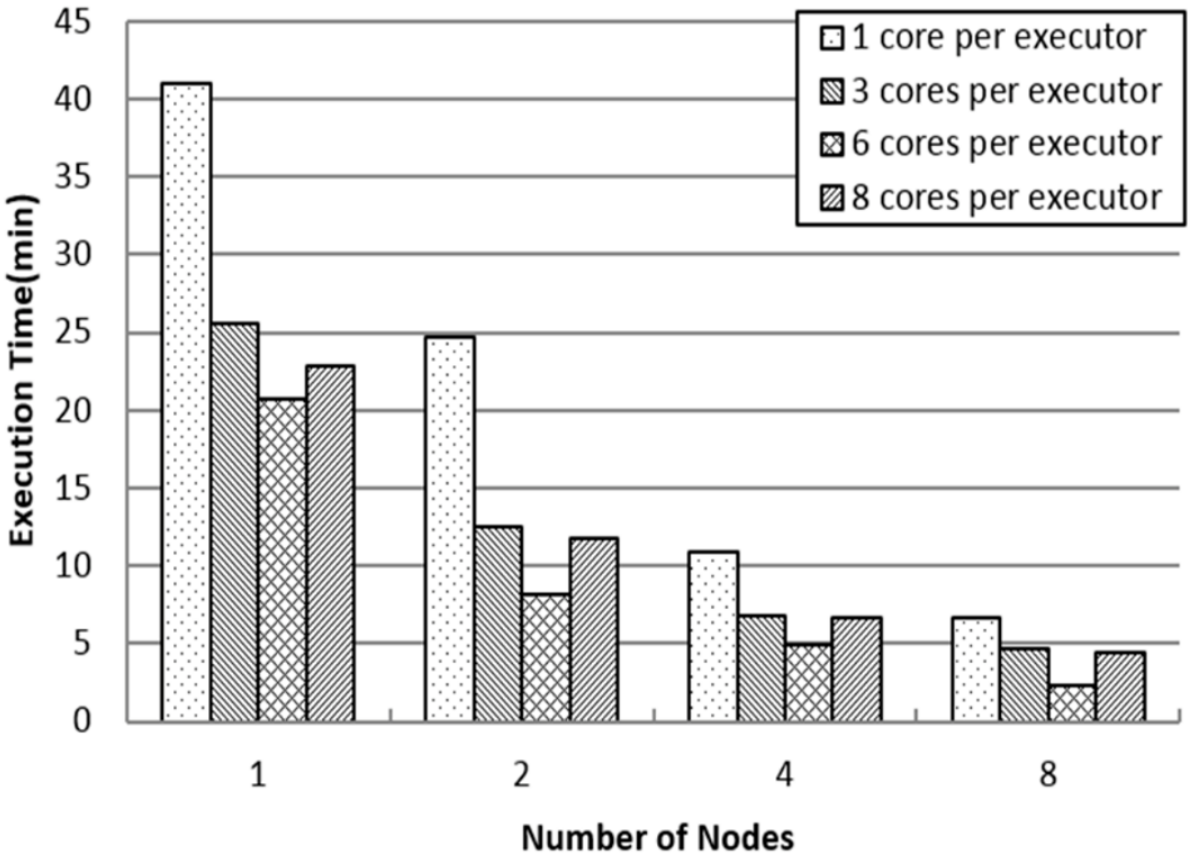
Publikacja	Porównane technologie	Wykorzystywana moc obliczeniowa	Analizowany wolumen danych	Liczba analizowanych rekordów
(Bao i inni, 2023)	Apache Spark (GeoMesa versus własne podejście)	3×(16 vCPU, 32GB RAM)	Brak danych	846 mln – 2.8 mld
(Kim, Hoang, Yu i Kanwar, 2023)	MongoDB, Couchbase, GeoMesa-Accumulo	3×(4 vCPU, 16GB RAM)	4 – 40 GB	26 tys. – 13 mln
(Zhang i inni, 2022)	Apache Sedona, Geoserver, PostGIS	48 vCPU, 96GB RAM	1 GB	94 tys. – 5.6 mln
(Zeidan i Vo, 2022)	LocationSpark, Simba, STARK, GeoSpark, SpPart_kNN	50×(5 vCPU, 32GB RAM)	38 MB – 142 GB	119 tys. – 3.78 mld
(Shin, Lee i Kwon, 2022)	GeoSpark (S3, HDFS, MongoDB)	6×(8 vCPU, 30GB RAM)	5 MB – 39 GB	7.3 tys. – 1 mld
(Li, Yu, Yuan i Qin, 2022)	SpatialHadoop	5×(8 vCPU, 16GB RAM)	379 MB – 10 GB	10 mln – 100 mln
(Baig, Teng, Kong i Wang, 2021)	SpatialHadoop (ST-Hadoop), Apache Flink (Spear, MobyDick)	20×(4 vCPU, 16GB RAM)	Brak danych	0.7 mln – 135 mln
(Haynes, Mitchell i Shook, 2020)	Apache Spark (GeoTrellis), SciDB, PostGIS	3×(24 vCPU, 128GB RAM)	5 MB – 1.7 GB (wektorowe)	49 – 64 tys (wektorowe), 18 mln – 1.69 mld pikseli (rastrowe)

Publikacja	Porównane technologie	Wykorzystywana moc obliczeniowa	Analizowany wolumen danych	Liczba analizowanych rekordów
(Alam M. M., 2019)	SpatialHadoop, GeoSpark, Apache Ignite (SpatialIgnite)	8×(16 CPU, 16GB RAM)	Brak danych	5.9 tys. – 4.1 mln
(Pandey, Kipf, Neumann i Kemper, 2018)	Apache Spark (GeoSpark, LocationSpark, Magellan, Simba, SpatialSpark)	16×(32 vCPU, 244GB RAM)	5 – 19 GB	70 mln – 200 mln
(Huang, Chen, Wan i Peng, 2017)	PostGIS, GeoSpark SQL, ESRI Spatial Framework for Hadoop	4 vCPU, 8GB RAM	723 – 929 MB	122 tys. – 5 mln
(Zhang, Zhou, Liu, Du i Ye, 2016)	SpatialHadoop, SpatialSpark	8×(6 vCPU, 6GB RAM)	77 MB – 62 GB	13 tys. – 164 mln

Analizując przedstawione powyżej publikacje można zauważyć szeroki zakres porównywanych technologii. Najwięcej badań, szczególnie w ostatnich latach, opiera się o Apache Spark. Liczne są też rozwiązania autorskie.

Wykorzystywana moc obliczeniowa jest bardzo różnorodna. Od 4 wirtualnych wątków procesora (vCPU) i 8GB RAM w lokalnych wersjach Spark/Hadoop, do klastrów składających się z 250 vCPU i 1.6 TB RAM. W tym miejscu warto wspomnieć, że w niniejszej pracy i badaniach autora rozprawy doktorskiej wykorzystano 512 vCPU i 2 TB pamięci RAM oraz analizowano 423,5 mld rekordów.

Częstym tematem prac badawczych jest kwestia podziału mocy obliczeniowych na maszyny wykonujące obliczenia (*worker*). Zależy to od konkretnego zadania oraz danych, ale autorzy są zgodni co do wyboru wartości pomiędzy 3, a 8 wątków procesora w przypadku obliczeń z użyciem Spark/Hadoop. Na rysunku 7 zaprezentowano przykład wyników eksperymentów dotyczących tego zagadnienia.



Rysunek 7. Porównanie wydajności analizy w zależności od liczby węzłów oraz liczby procesorów na węzeł w pracy (Zhang, Zhou, Liu, Du i Ye, 2016)

W tabeli 4 przedstawiono zestawienie badanych algorytmów w poszczególnych badaniach wraz z krótkim opisem specyfiki danych oraz kluczowych wniosków uzyskanych w trakcie badań.

Tabela 4. Zestawienie badanych algorytmów, charakterystyki danych oraz głównych wniosków w analizowanych badaniach porównawczych SBD (opracowanie własne)

Publikacja	Badane algorytmy	Opis danych	Główny wniosek
(Bao i inni, 2023)	Utworzenie map gęstości w siatce 1×1 km	Dane wektorowe (trajektorie GPS taksówek w Pekinie)	Własny system przechowywania trajektorii wydajniejszy niż GeoMesa
(Kim, Hoang, Yu i Kanwar, 2023)	Złączenie przestrzenne	Dane wektorowe (podział administracyjny w Japonii, trasy busów – sztucznie powiększone)	Stworzono zbiór danych testowych dla NoSQL, potwierdzono przydatność na MongoDB, Couchbase, Accumulo

Publikacja	Badane algorytmy	Opis danych	Główny wniosek
(Zhang i inni, 2022)	Tworzenie kafelków wektorowych	Dane topograficzne Pekinu – POI, cieki, drogi, budynki, wegetacja	Apache Sedona znacznie przewyższyła wydajnościowo PostGIS i GeoServer
(Zeidan i Vo, 2022)	K najbliższych sąsiadów	Dane wektorowe (POI, mobilność – busy, taxi w Nowym Yorku)	Autorskie rozwiązanie jest szybsze i oszczędniejsze niż istniejące systemy
(Shin, Lee i Kwon, 2022)	K najbliższych sąsiadów, selekcja przestrzenna w obszarze koła i prostokąta	Dane wektorowe, POI (prawdziwe i generowane)	MongoDB wolniejsze we wszystkich przypadkach; S3 ≈ HDFS
(Li, Yu, Yuan i Qin, 2022)	Złączenie czasowo-przestrzenne z różnymi technikami indeksowania	Dane wektorowe, trajektorie (prawdziwe i generowane)	Algorytm STPRQ najszybszy względem innych podejść
(Baig, Teng, Kong i Wang, 2021)	Czasowo-przestrzenne zapytania – zakres, k najbliższych sąsiadów	Dane z projektu Array of Things (sensory, dane dzienne–miesięczne)	Autorski algorytm SPEAR wydajniejszy niż ST-Hadoop i MobyDick
(Haynes, Mitchell i Shook, 2020)	Zliczanie pikseli, reklasyfikacja, dodawanie rastrów, średnia w oknie 3×3, statystyki w poligonach	Dane rastrowe (pokrycie terenu USA), dane wektorowe (podział administracyjny USA)	Geotrellis najlepszy w algebrze rastrów, PostGIS w statystykach poligonów
(Alam M. M., 2019)	Złączenia przestrzenne, analiza (bufor, długość, otoczka wypukła, powierzchnia)	Dane wektorowe (POI, granice, cieki, drogi)	Zdefiniowano zestaw operacji i danych do porównania wydajności systemów SBD
(Pandey, Kipf, Neumann i Kemper, 2018)	K najbliższych sąsiadów, złączenia przestrzenne	Dane OSM – POI, drogi, budynki	GeoSpark najlepszy w większości testów
(Huang, Chen, Wan i Peng, 2017)	K najbliższych sąsiadów, złączenia przestrzenne, zapytania atrybutowe	Dane wektorowe (symulowane trajektorie tajfunów – punktowe; klasyfikacja	GeoSpark lepszy niż PostGIS w większości testów

Publikacja	Badane algorytmy	Opis danych	Główny wniosek
		pokrycia terenu – poligonowe)	
(Zhang, Zhou, Liu, Du i Ye, 2016)	Złączenia przestrzenne	Dane wektorowe (OSM i TIGER)	Apache Spark znacznie szybszy niż Hadoop; przewaga rośnie z rozmiarem danych

Przenalizowane badania dotyczą zagadnień takich jak algorytmy indeksowania danych przestrzennych, sposób ich przechowywania w systemach rozproszonych oraz próby optymalizacji wybranych algorytmów (w szczególności K najbliższych sąsiadów (Cover i Hart, 1967) czy złączeń przestrzennych).

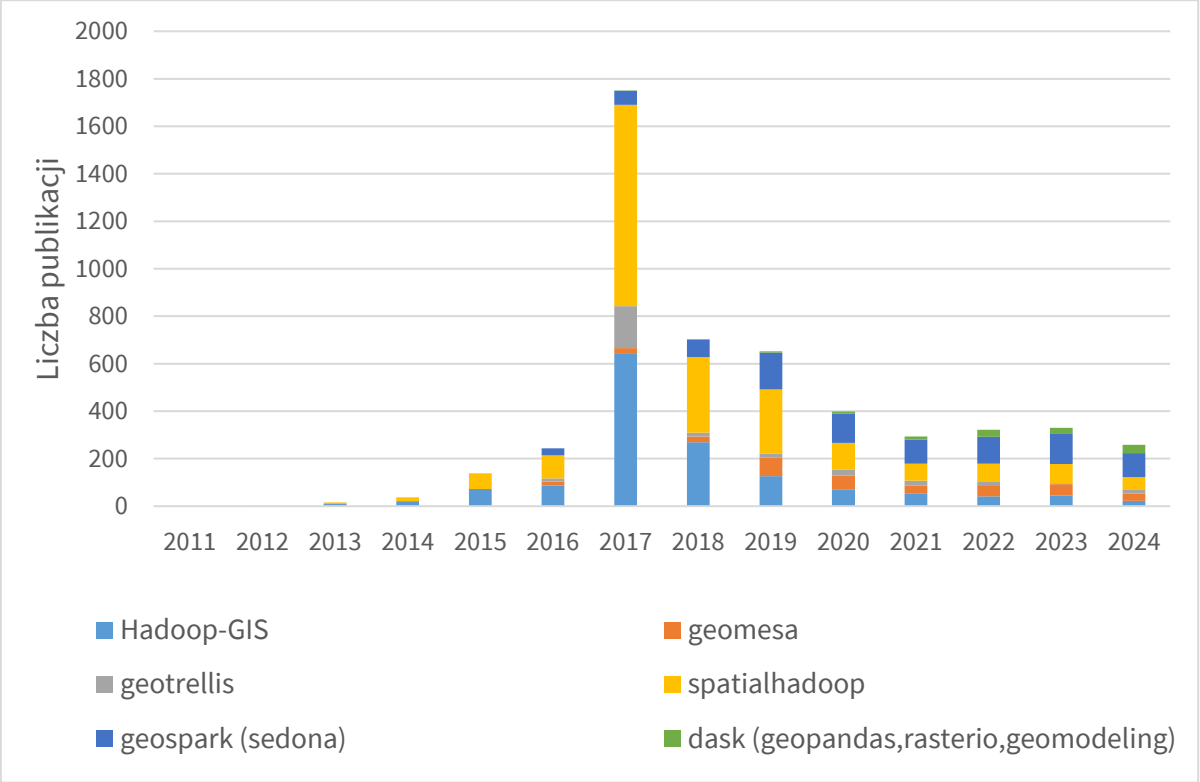
Najczęściej wykorzystywanym w badaniach typem danych były dane wektorowe (w szczególności dane topograficzne oraz przechowujące trajektorie pojazdów). Wielkość badanych zbiorów danych była bardzo zróżnicowana. Ich rozpiętość wynosi od pojedynczych megabajtów (tysiące rekordów) w pracy (Zhang, Zhou, Liu, Du i Ye, 2016) do ok. 150 GB (2-4 miliardy rekordów) w (Zeidan i Vo, 2022).

W badaniach (Kim, Hoang, Yu i Kanwar, 2023), (Shin, Lee i Kwon, 2022) oraz (Li, Yu, Yuan i Qin, 2022) sztucznie wygenerowano dane na podstawie mniejszych zbiorów źródłowych. Niemal wszystkie badania obejmowały analizę skalowania obliczeń w zależności od wielkości danych.

W zakresie analizowanych prac nie zidentyfikowano badań na danych rzędu terabajtów. Może to być spowodowane czasochłonnością takich obliczeń, kosztem a także utrudnionym dostępem do danych przestrzennych tej skali. Pomimo, iż część badań postulowała utworzenie zbioru porównawczego, możliwego do wykorzystania w innych pracach, nie znaleziono dwóch prac wykorzystujących ten sam zbiór danych. Może to wskazywać, że w tematyce SBD nie istnieją jeszcze zbiory referencyjne umożliwiające rzetelne porównanie metod obliczeniowych przez różnych badaczy tak, jak ma to miejsce na dużą skalę np. w przypadku algorytmów sztucznej inteligencji (przykładem mogą być zbiory danych uczących DIOR (Li, Wan, Cheng, Meng i Han, 2020) czy DOTA (Xia i inni, 2018)).

3. Zastosowanie SBD w badaniach naukowych

Rozwiązania SBD cieszą się zainteresowaniem w środowisku naukowym (rysunek 8). Podobnie, jak w przypadku samego terminu „*spatial big data*”, tak i w przypadku poszczególnych narzędzi rekordowy był rok 2017, gdzie nastąpiło bardzo duże zainteresowaniem oprogramowaniem SpatialHadoop w latach kolejnych popularność tego narzędzia jednak już regularnie spadała ustępując miejsca pakietom oprogramowania związanymi z Apache Spark (Apache Sedona, GeoMesa) czy z Dask. Pomimo ogólnego wzrostu liczby publikacji nawiązujących do SBD rok do roku na przestrzeni ostatnich 4 lat, same narzędzia SBD utrzymują w tych latach zrównoważony poziom zainteresowania.

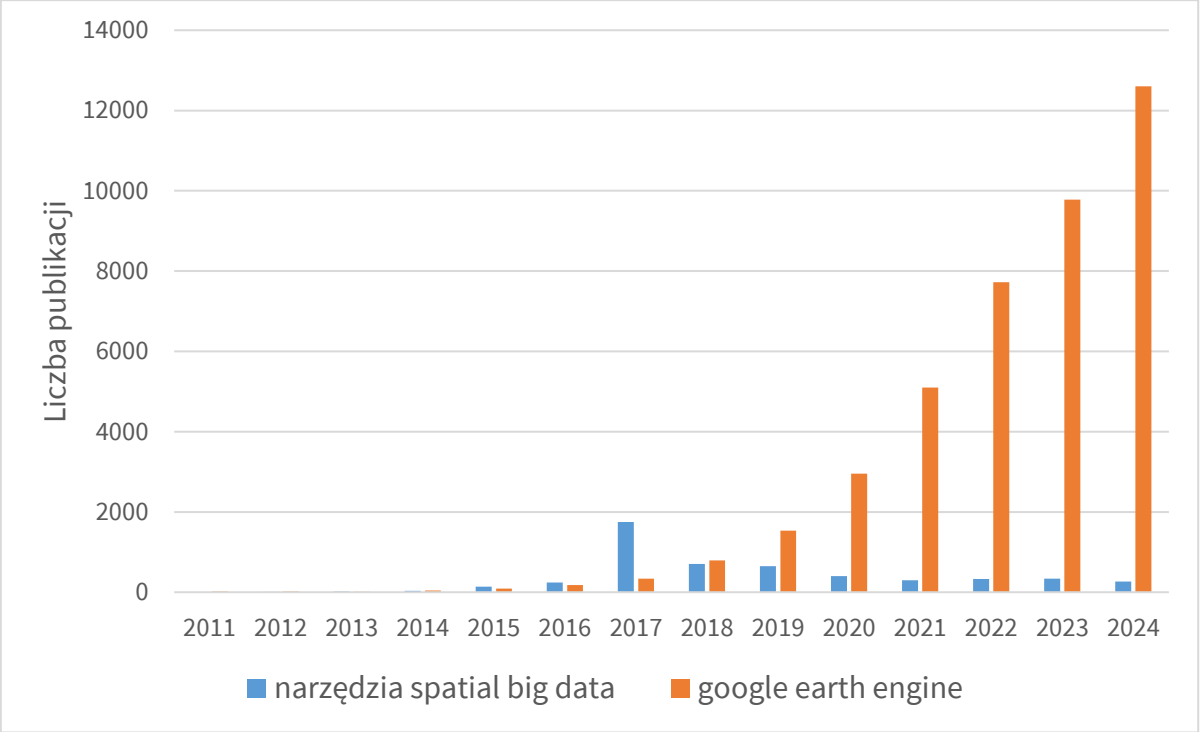


Rysunek 8. Wybrane narzędzia Spatial Big Data występujące w tekście prac naukowych dostępnych w bazie Dimensions w latach 2011-2024 (opracowanie własne)

W dalszym ciągu największym zainteresowaniem wśród naukowców cieszą się pakiety oparte o Spark oraz Hadoop, pomimo tego, że część z nich nie jest już regularnie wspierana.

W przypadku rozwiązań komercyjnych SBD, których największym reprezentantem jest oprogramowanie Google Earth Engine (Loukili, Lakhrissi i Ali, 2022) sytuacja wygląda inaczej, gdyż odniesienia do tego oprogramowania występują częściej w ujęciu rok do roku

i przewyższają liczbę odniesień do wszystkich wcześniej porównywanych narzędzi, z wyjątkiem roku 2017 (rysunek 9). Trzeba w tym przypadku wziąć pod uwagę, że wymienione powyżej narzędzia mają charakter specjalistyczny i dedykowane są wyłącznie analizom SBD, podczas gdy GEE jest przystosowane również do klasycznych analiz GIS i posiada pełne wsparcie dla początkującego użytkownika.



Rysunek 9. Narzędzia Spatial Big Data (Hadoop-GIS, geomesa, geotrellis, spatialhadoop, geospark (sedona) oraz dask (geopandas, rasterio, geomodelling)) oraz fraza Google Earth Engine występujące w tekście prac naukowych dostępnych w bazie danych Dimensions w latach 2011-2024 (opracowanie własne)

Tematyka SBD cieszy się dużą popularnością wśród badaczy, jednak badania porównawcze w tym zakresie nie przedstawiają wprost analiz wydajności tych narzędzi w odniesieniu do bardzo dużych zbiorów danych przestrzennych.

W dalszej kolejności pracy nad niniejszą rozprawą przeprowadzono więc przegląd literatury w celu odnalezienia prac, w których badano właśnie takie zbiory. W tym wypadku brano pod uwagę jedynie zbiory, których wielkość wynosiła przynajmniej kilkadziesiąt gigabajtów, bądź analizie podlegały dziesiątki miliardów punktów pomiarowych. W wielu pracach autorzy niestety nie precyzowali dokładnej wielkości wykorzystywanych zbiorów, co wymagało oszacowania ich rozmiaru na podstawie dostępnych informacji opisowych. Tam, gdzie było to możliwe pozyskano też informacje o zastosowanej technologii oraz wykorzystanych mocach

obliczeniowych. Poniżej przedstawiono wyniki wykonanej analizy. W tabeli 5 oraz tabeli 6 zaprezentowano kluczowe informacje dotyczące tych badań.

Tabela 5. Zestawienie metadanych dotyczących badań dotyczących analiz SBD (opracowanie własne)

Lp.	Publikacja	Wykorzystane technologie SBD	Wykorzystana moc obliczeniowa	Skala
1	(Hansen i inni, 2013)	Google Cloud	Nie podano	Globalna
2	(Pekel, Cottam, Gorelick i Belward, 2016)	Google Earth Engine	Nie podano	Globalna
3	(Hu i inni, 2017)	Apache Hadoop	36 serwerów (16 rdzeni CPU i 16 GB RAM każdy)	Globalna
4	(Esch i inni, 2017)	Apache Hadoop	28 serwerów (4 rdzenie CPU i 32 GB RAM każdy)	Globalna
5	(Kroodsma i inni, 2018)	Nie podano	Nie podano	Globalna
6	(Tang i inni, 2018)	Microsoft HPC Pack	30 serwerów (4 rdzenie CPU każdy)	Globalna
7	(Deibe, Amor i Doallo, 2020)	Apache Spark	16 serwerów (32 rdzenie CPU i 64 GB RAM każdy)	Regionalna
8	(Raffa i inni, 2021)	Infrastruktura CINECEA	60 serwerów (łącznie 2160 rdzeni)	Krajowa
9	(Kissling i inni, 2023)	DASK	11 maszyn wirtualnych (2 rdzenie i 32 lub 64 GB RAM każda)	Krajowa
10	(Teijeiro Paredes, Amor López, Buján, Richter, i Döllner, 2024)	Apache Spark	13 serwerów (16 rdzeni i 64 GB RAM każdy)	Regionalna

Tabela 6. Zestawienie opisów danych i przeprowadzonych analiz w badaniach dotyczących analiz SBD (opracowanie własne)

Lp.	Publikacja	Opis danych	Przeprowadzone operacje
1	(Hansen i inni, 2013)	Dane satelitarne Landsat obrazujące tereny leśne na całej plancie (rozdzielczość 30 m)	Analiza zmian zadrzewienia w latach 2000 - 2012
2	(Pekel, Cottam, Gorelick i Belward, 2016)	Dane satelitarne Landsat obrazujące 99.95% lądu na Ziemi o łącznej wielkości **1823 TB** (rozdzielczość 30 m)	Analiza zmian występowania wód powierzchniowych w latach 1984 - 2015
3	(Hu i inni, 2017)	Dzienne dane klimatyczne MERRA dla obszaru USA o łącznej wielkości **100 GB**	Detekcja anomalii w danych klimatycznych w latach 1979 - 2013
4	(Esch i inni, 2017)	Satelitarne dane radarowe TerraSAR-X i TanDEM-X dla obszaru całego świata o łącznej wielkości ok. **400 TB**	Wytworzenie globalnej mapy wpływu miast na środowisko
5	(Kroodsma i inni, 2018)	**22 miliardy punktów** pomiarowych lokalizacji jednostek wodnych systemu automatycznej identyfikacji	Budowa modelu opartego o konwolucyjną sieć neuronową do analizy globalnej charakterystyki operowania kutrów rybackich
6	(Tang i inni, 2018)	Dane wysokościowe SRTM (rozdzielczość 30 m) dla całej planety o wielkości około **315 GB**)	Estymacja światowej biomasy namorzyn
7	(Deibe, Amor i Doallo, 2020)	Dane z lotniczego skanowania laserowego dla regionu Galicja w Hiszpanii o wielkości **802 GB**	Generowanie Numerycznego Modelu Terenu na podstawie chmury punktów
8	(Raffa i inni, 2021)	Dane klimatyczne ERA5 dla obszaru całych Włoch	Zwiększanie rozdzielczości wejściowych danych do ok. 2.2 km. Wielkość wyjściowego zbioru danych to 78 TB
9	(Kissling i inni, 2023)	Dane z lotniczego skanowania laserowego dla terenu całej Holandii o wielkości **16 TB** (ok. **700 miliardów punktów** pomiarowych)	Generowanie tematycznych warstw rastrowych na podstawie sklasyfikowanej chmury punktów
10	(Teijeiro Paredes, Amor López, Buján, Richter, i Döllner, 2024)	Dane z lotniczego skanowania laserowego dla regionu Navarra w Hiszpanii o wielkości **75 GB** (**dwa miliardy punktów**)	Filtracja punktów reprezentujących grunt

Analiza literatury wykazała, że w podanym powyżej kontekście naukowym najczęściej wykorzystuje się metody SBD do analizy danych rastrowych, chmur punktów oraz danych klimatycznych/pogodowych. Jest to zapewne spowodowane faktem, że te zbiory danych w swojej naturze cechują się wysoką rozdzielczością przestrzenną oraz czasową, a także szerokim zakresem dostępności. Metody te wykorzystywane są jednak także w analizie danych wektorowych, szczególnie tych zbieranych metodą *crowdsourcingu* (np. dane FCD).

Można zauważyć dużą różnorodność narzędzi wykorzystywanych w omawianych badaniach — od Apache Hadoop i Spark, przez Dask oraz rozwiązania komercyjne, takie jak Google Earth Engine, aż po dedykowane infrastruktury informatyczne. Należy pamiętać, że w opracowaniach naukowych dane oraz metody ich przetwarzania nie zawsze stanowią główny przedmiot zainteresowania badaczy, dlatego w wielu pracach, w których potencjalnie mogłyby zostać zastosowane metody SBD do analizy bardzo dużych zbiorów danych przestrzennych, nie jest to wyraźnie odnotowywane.

Najczęściej dane analizowane są w skali globalnej, dla całego świata. Znaleziono jednak przykłady badań w skali makroregionu bądź kraju. W takich wypadkach były to najczęściej dane pochodzące z lotniczego skanowania laserowego.

Przetwarzane wolumeny zbiorów danych oscylują pomiędzy kilkudziesięcioma gigabajtami, a tysiącami terabajtów. Należy jednak podkreślić, że posługiwanie się wyłącznie miarą zajętości przestrzeni dyskowej nie stanowi w pełni miarodajnej podstawy porównań pomiędzy różnymi typami danych przestrzennych, takimi jak dane rastrowe, wektorowe czy chmury punktów. Wynika to z odmiennych modeli reprezentacji informacji, różnego stopnia złożoności geometrycznej, zróżnicowanej struktury atrybutowej oraz odmiennej podatności na kompresję i metody organizacji zapisu. Mimo tych ograniczeń, w analizowanych pracach naukowych pojemność zbioru wyrażona w jednostkach pamięci masowej była bardzo często jedyną wskazywaną miarą skali danych.

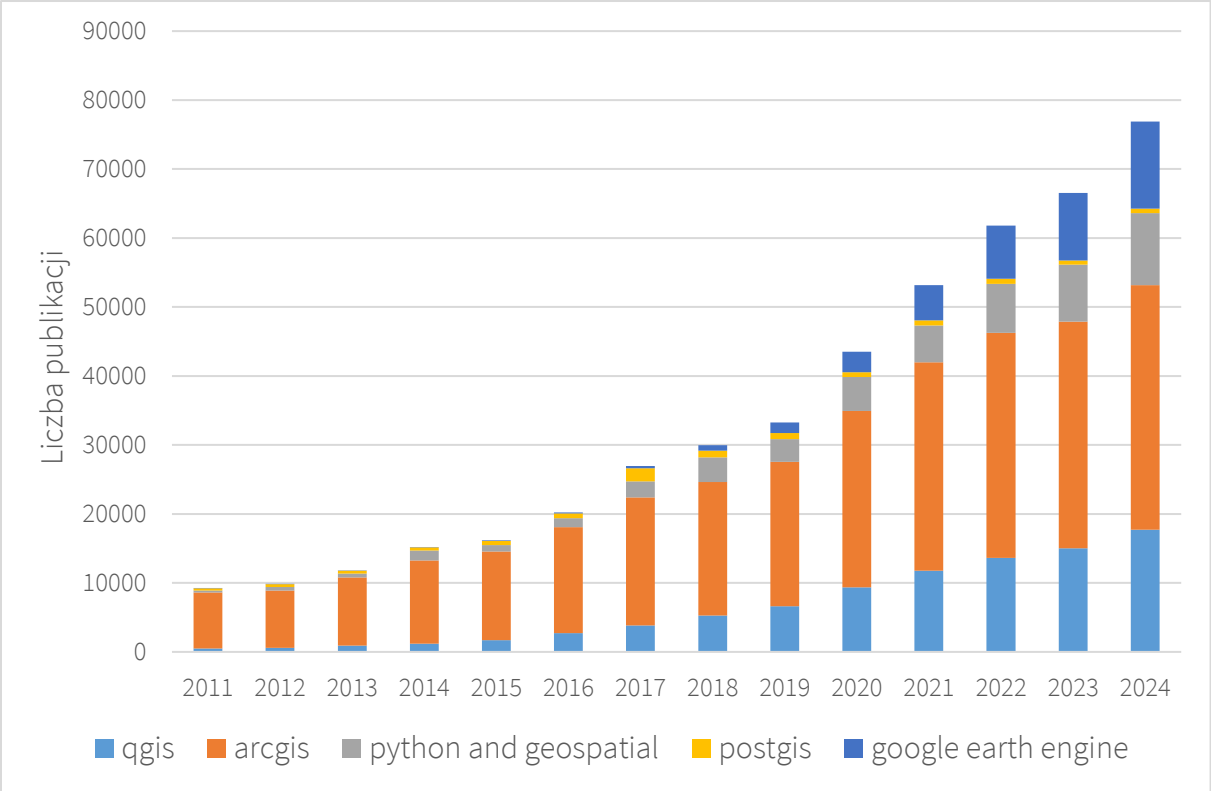
Należy zauważyć, że metody SBD weszły do zbioru narzędzi obliczeniowych wykorzystywanych przez naukowców analizujących dane przestrzenne w dużej skali, jednak wcześniejsza analiza ilościowa literatury sugeruje, że nie są i prawdopodobnie nie będą to dominujące narzędzia obliczeniowe.

4. Narzędzia SBD a GIS

Kluczowe dla realizacji celów pracy jest zrozumienie relacji pomiędzy stosowanymi na szeroką skalę narzędziami analizy GIS, a metodami SBD. Przykładami klasycznych narzędzi GIS jest otwarte oprogramowanie QGIS, komercyjne środowiska ArcGIS firmy Esri i GeoMedia[10] firmy Hexagon, systemy zarządzania bazami danych przestrzennych (np. PostGIS[11] czy Oracle Spatial Database[12]). Technologia GIS i funkcje GIS są też dostępne poprzez biblioteki dla języków programowania. Przykładem może być np. język Python[13] z bibliotekami geopandas[14], shapely[15], laspy[16] i język R[17] z bibliotekami sf[18], terra[19] czy stars[20]. Ten rozdział skupia się na analizie wykorzystania tak rozumianych zarówno klasycznych narzędzi GIS jaki i języków programowania z bibliotekami GIS, względem narzędzi SBD oraz na określeniu cech odróżniających oba podejścia.

Popularność klasycznych narzędzi desktop GIS w badaniach naukowych zdecydowanie przewyższa popularność zarówno narzędzi SBD opartych o *open source*, jak i tych komercyjnych. Na rysunku 10 przedstawiono zestawienie wystąpień popularnych narzędzi GIS w pracach naukowych względem wystąpień frazy Google Earth Engine jako najczęściej występującego w pracach naukowych narzędzia SBD.

[10] https://hexagon.com/products/geomedia (data dostępu 15.12.2025)
[11] https://postgis.net (data dostępu 15.12.2025)
[12] https://www.oracle.com/pl/database/spatial-database/ (data dostępu 15.12.2025)
[13] https://www.python.org (data dostępu 15.12.2025)
[14] https://geopandas.org (data dostępu 15.12.2025)
[15] https://shapely.readthedocs.io (data dostępu 15.12.2025)
[16] https://laspy.readthedocs.io (data dostępu 15.12.2025)
[17] https://www.r-project.org (data dostępu 15.12.2025)
[18] https://r-spatial.github.io/sf (data dostępu 15.12.2025)
[19] https://rspatial.github.io/terra (data dostępu 15.12.2025)
[20] https://r-spatial.github.io/stars (data dostępu 15.12.2025)

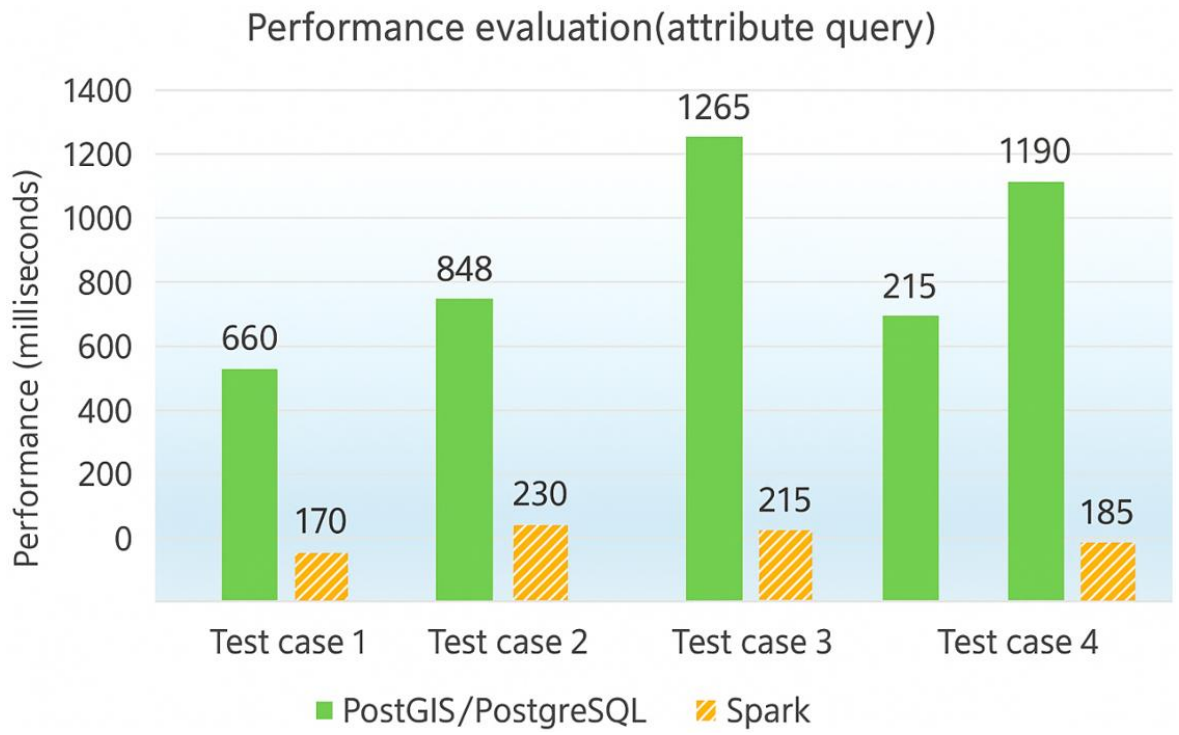


Rysunek 10. Występowanie słów związanych z narzędziami GIS w tekście prac naukowych dostępnych w bazie Dimensions w latach 2011-2024 (opracowanie własne)

Pomimo rosnącej popularności GEE zauważyć można zdecydowaną dominację klasycznych narzędzi GIS, w szczególności tych klasy desktop GIS. Liczba wystąpień narzędzia firmy Google jest porównywalna do wystąpień dla języka Python w połączeniu ze słowem „*geospatial*". Obie frazy zyskują również na popularności w ostatnich latach.

Jednak również popularność narzędzi desktop GIS stale rośnie i utrzymuje znaczną przewagę względem innych analizowanych technologii, co potwierdza obserwacje, że narzędzia *spatial big data* nie zdominowały tej dziedziny, a wiele zadań jest osiągalnych z wykorzystaniem klasycznych narzędzi i ograniczonych mocy obliczeniowych.

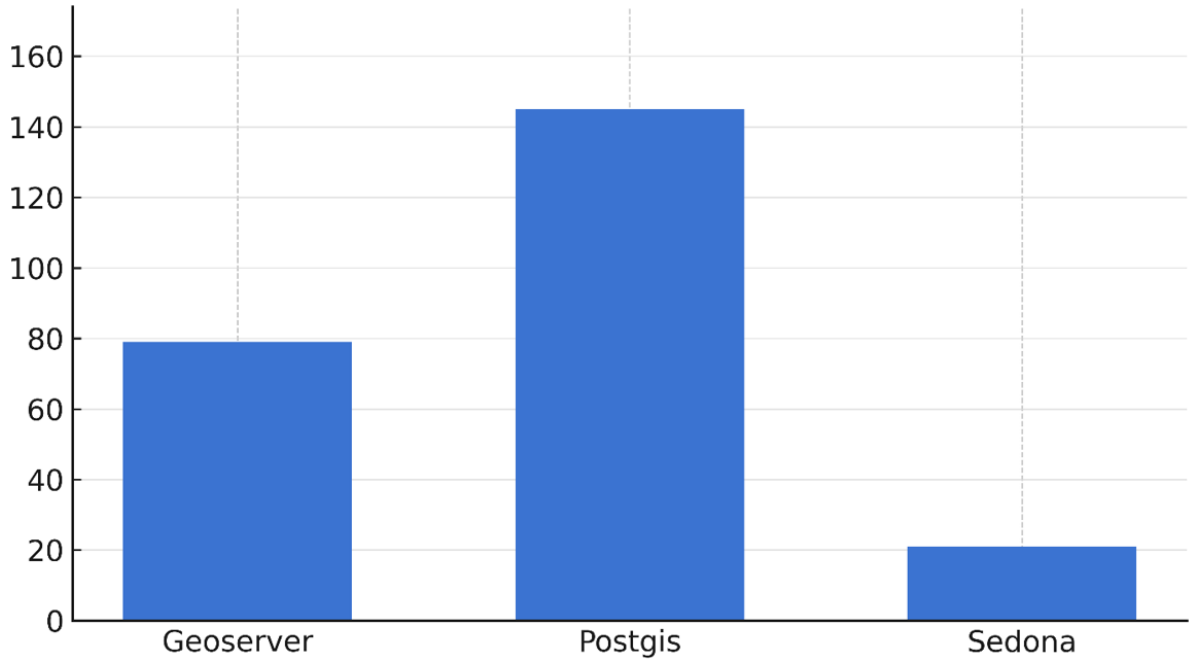
Dostępnych jest niewiele publikacji porównujących narzędzia *spatial big data* i klasyczne rozwiązania GIS pod względem wydajności i przydatności dla analizy konkretnych zbiorów danych czy potrzebnych mocy obliczeniowych. Można jednak znaleźć kilka interesujących przykładów. Pierwszym z nich jest (Huang, Chen, Wan i Peng, 2017), gdzie porównywana jest wydajność bazy PostGIS względem pakietu GeoSpark w różnych przypadkach testowych (rysunek 11).



Rysunek 11. Wyniki badania porównawczego wydajności zapytań atrybutowych PostGIS względem GeoSpark (Huang, Chen, Wan i Peng, 2017)

W powyższej pracy zbadano wydajność PostGIS oraz GeoSpark w zapytaniach atrybutowych, k-najbliższych sąsiadów, geometrii zawierających punkt, geometrii wewnątrz prostokąta, relacji topologicznych i złączeń przestrzennych. Badania przeprowadzono z użyciem symulowanego zbioru punktowego zawierającego 4.9 mln punktów (929 MB) oraz poligonowego zawierającego blisko 129 tysięcy punktów (723 MB). W większości wypadków zapytania w GeoSpark były szybsze niż te z PostGIS. Oba systemy testowane były jednak w różnych środowiskach, a także nie jest jasne jaka liczba maszyn została wykorzystana w obliczeniach Spark.

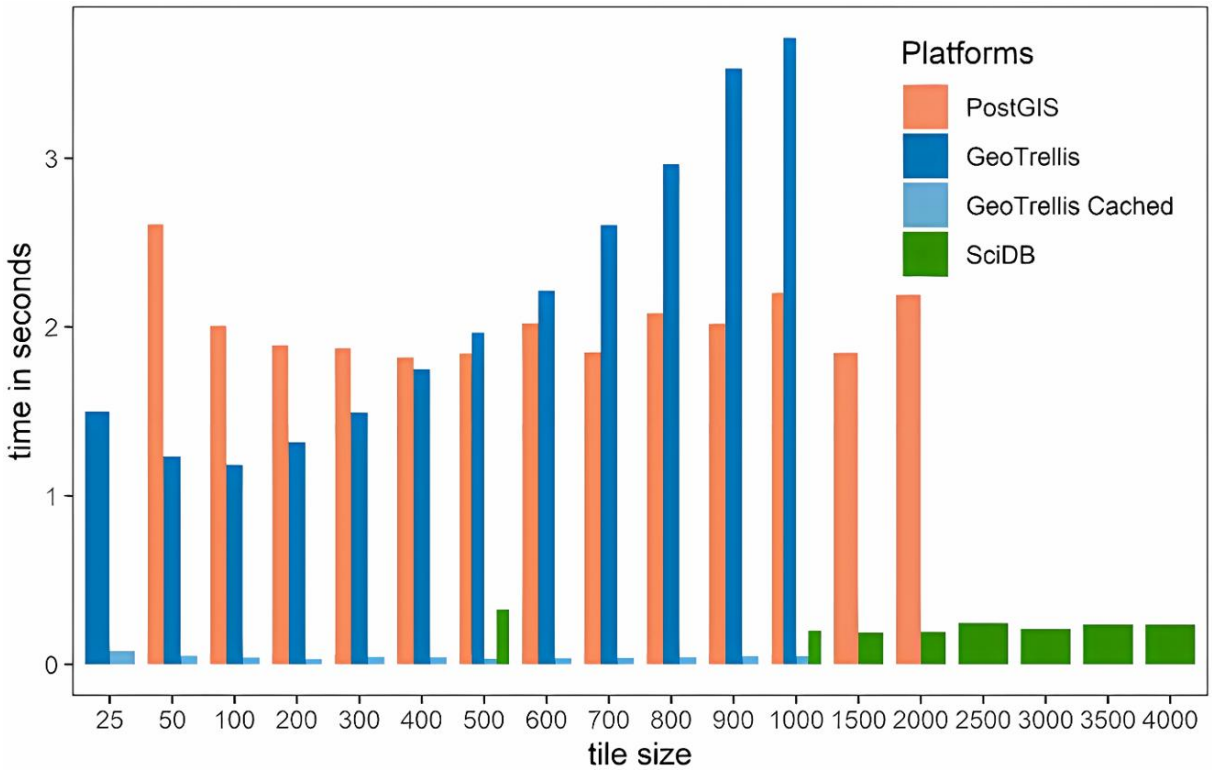
Kolejnym przykładem porównania jest (Zhang i inni, 2022), gdzie wydajność bazy PostGIS oraz oprogramowania Geoserver porównano z wydajnością uzyskaną z użyciem pakietu Apache Sedona (rysunek 12).



Rysunek 12. Czas (w minutach) generowania kafelków wektorowych z użyciem trzech środowisk: Geoserver, PostGIS oraz Apache Sedona (w minutach). Na podstawie (Zhang i inni, 2022)

Powyższe badanie sprawdzało czas generowania kafelków wektorowych na podstawie zbioru danych o wielkości ok. 1GB. Oprogramowanie Apache Sedona uzyskało najlepszy wynik ze wszystkich analizowanych rozwiązań. W tym wypadku wskazano parametry serwera, na którym wykonano porównanie z użyciem wszystkich trzech rozwiązań. Była to maszyna wirtualna z 48. rdzeniami procesora i 96. GB pamięci RAM. Nie podano jednak konfiguracji poszczególnych aplikacji i nie wskazano, czy wszystkie badane aplikacje wykorzystały dostępne zasoby.

Zidentyfikowano jedno opracowanie porównujące wydajność narzędzi GIS z SBD w odniesieniu do danych rastrowych (Haynes, Mitchell i Shook, 2020). Baza danych PostGIS jest w nim porównywana pod względem wydajności do rozwiązań SBD GeoTrellis oraz SciDB (Cudré-Mauroux i inni, 2009). Wyniki jednego z eksperymentów tego badania przedstawiono na rysunku 13.



Rysunek 13. Wyniki badania porównawczego środowisk PostGIS, GeoTrellis oraz SciDB (operacja zliczenia liczby pikseli) (Haynes, Mitchell i Shook, 2020)

W ramach badania przeanalizowano wydajność narzędzi w operacjach zliczenia pikseli, sumy rastrów, reklasyfikacji, obliczenia średniej w oknie 3x3 pikseli oraz obliczenia statystyk w poligonie. Analizowano obrazy składające się z od 18 milionów do 1.69 miliarda pikseli. Autorzy udowodnili, że PostGIS poradził sobie najlepiej w operacji zliczania statystyk w poligonach, a GeoTrellis osiągał najlepszą wydajność w prostszych operacjach, nie wymagających łączenia przestrzennego danych wektorowych i rastrowych.

Każda z przeanalizowanych prac opiera się o stosunkowo mały zbiór danych i nie weryfikuje wydajności w zależności od wielkości zbioru. Nie zidentyfikowano prac porównujących inne rodzaje oprogramowania GIS z metodami SBD.

Praca przeglądowa z 2025 roku „Remote Sensing and Geospatial Analysis in the Big Data Era: A Survey" (Dritsas i Trigka, 2025) omawia próby określenia cech tradycyjnych metod GIS w porównaniu z metodami SBD. W tabeli 7 przedstawiono zestawienie aspektów rozróżniających metody tradycyjne od metod SBD opracowane przez autorów tej publikacji.

Tabela 7. Porównanie poszczególnych aspektów przetwarzania danych przestrzennych w rozróżnieniu na podejście tradycyjne i oparte o metody SBD. (Dritsas i Trigka, 2025)

Aspekt	Metody tradycyjne	Metody z wykorzystaniem *big data*
Wolumen danych	**Ograniczone zbiory danych** zbierane okresowo z pojedynczych źródeł, takich jak satelity czy zdjęcia lotnicze.	Ogromne, heterogeniczne zbiory danych z satelitów, dronów (UAV), czujników IoT i mediów społecznościowych, osiągające **skalę petabajtów**.
Przetwarzanie danych	**Ręczne, pracochłonne** procesy przy użyciu desktopowego **oprogramowania GIS**.	Zautomatyzowane procesy z wykorzystaniem **platform chmurowych** (np. Google Earth Engine) oraz algorytmów AI/ML.
Rozdzielczość czasowa	**Stałe interwały** wynikające z częstotliwości przelotów satelitów (np. dziennie, tygodniowo).	**Strumienie danych** w niemal rzeczywistym czasie z sieci IoT i konstelacji satelitarnych (np. Planet Labs, Sentinel-2).
Integracja danych	Ograniczona, często ręczna integracja danych z różnych źródeł.	Zaawansowane techniki fuzji danych z wykorzystaniem algorytmów uczenia maszynowego.
Techniki analityczne	Modele oparte na regułach, techniki statystyczne oraz podstawowa klasyfikacja obrazów.	Podejścia oparte na AI (np. CNN do analizy obrazów, RNN do danych czasowych, transfer learning do zadań specjalistycznych).
Zastosowania	Mapowanie, statyczna klasyfikacja pokrycia terenu, analiza trendów historycznych.	Analizy predykcyjne, monitoring w czasie rzeczywistym, reagowanie na katastrofy, planowanie urbanistyczne, rolnictwo precyzyjne, modelowanie zmian klimatycznych.
Skalowalność	**Ograniczona skalowalność** ze względu na zależność od	**Skalowalne rozwiązania** dzięki chmurze obliczeniowej (AWS, Google Cloud) i rozproszonym systemom

Aspekt	Metody tradycyjne	Metody z wykorzystaniem *big data*
	lokalnego sprzętu i oprogramowania.	przechowywania danych (HDFS, Apache Spark).
Dostępność	Ograniczony dostęp z powodu wysokich kosztów oprogramowania komercyjnego i niewystarczających zasobów obliczeniowych.	Zdemokratyzowany dostęp przez platformy *open source*, usługi chmurowe i inicjatywy udostępniania danych (np. Copernicus Open Access Hub).
Wyzwania	Ograniczenia związane z wolumenem danych, pojemnością magazynowania i ręcznymi procesami.	Wyzwania etyczne i techniczne, w tym prywatność danych, interoperacyjność oraz stronniczość modeli AI.

Tak zdefiniowane aspekty różniące klasyczny GIS od metod SBD mogą pomóc w zrozumieniu różnic, ale nie w pełni oddają realną złożoność i niuanse obu podejść:

- Analizy przestrzenne w skali petabajtów nie są częste w wielu zadaniach analizy GIS nawet przy obecnie rosnących wolumenach danych. Tak duże analizy są też zwykle przeprowadzane z wykorzystaniem dedykowanych superkomputerów wykorzystujących autorskie oprogramowanie
- Klasyczne analizy GIS przeprowadzane są także z użyciem języków programowania takich jak Python czy R oraz wydajnych, relacyjnych baz danych (np. PostGIS). Na tym polu również wprowadza się wiele optymalizacji pozwalających na wydajne obliczenia nawet przy ograniczonych zasobach
- Zastosowanie rozwiązań rozproszonych, w szczególności tych *open source* jak Apache Spark może wiązać się ze znacznie większym stopniem skomplikowania i wysokim progiem wejścia merytorycznego w porównaniu z technologiami GIS
- Techniki szeroko rozumianej AI są z powodzeniem stosowane zarówno w klasycznych narzędziach Desktop GIS, jak i z użyciem pojedynczych maszyn obliczeniowych

- Lata rozwoju klasycznych metod GIS spowodowały, że wiele metod jest dobrze sprawdzonych i pewnych. Technologie SBD nie są tak dojrzałe i pomimo stosunkowo długiej obecności na rynku, wiele algorytmów wymaga ręcznej implementacji bądź optymalizacji
- Klasycznie rozumiany GIS coraz częściej przeplata się i integruje z metodami SBD (np. Integracja Spark w Esri ArcGIS).

5. Cyberinfrastruktury danych przestrzennych

Pojęcie cyberinfrastruktura danych przestrzennych (ang. *Geospatial Cyberinfrastructure* – GCI) odnosi się do złożonej i wysokowydajnej infrastruktury informatycznej wspierającej procesy pozyskiwania, zarządzania, udostępniania, analizy i wizualizacji danych przestrzennych na potrzeby wielu dziedzin nauki. Pojęcie to zostało wprowadzone wraz z rosnącą dostępnością informacji przestrzennej w roku 2005. Powstawanie specjalnych cyberinfrastruktur danych przestrzennych miało odpowiedzieć na problem wydajnej analizy coraz większych zbiorów danych nie tylko pod względem samej mocy obliczeniowej, ale także metod ich przetwarzania (algorytmów) i sposobów efektywnego współdzielenia oraz przechowywania informacji przestrzennej. Przyjmuje się, że elementem składowym GCI jest także społeczność naukowców, pracowników z sektora publicznego i prywatnego oraz innych interesariuszy, którzy decydują o kierunku rozwoju cyberinfrastruktury i ten rozwój wspierają swoją ekspercką wiedzą (Yang, Raskin, Goodchild i Gahegan, 2010).

Powyższa definicja GCI jest bardzo szeroka, a problematyka obliczeń typu SBD stanowi jedynie podzbiór zagadnień dotyczących GCI. Twórcy i zarządzający komercyjnymi oraz naukowymi infrastrukturami informatycznymi nazywają tego typu rozwiązania także w inny sposób np. jako platformy analizy danych przestrzennych, platformy informacji przestrzennej, platformy chmurowe, platformy SBD. Spełniają one cechy zdefiniowane dla GCI przez (Yang, Raskin, Goodchild i Gahegan, 2010) w różnym stopniu. Porównanie tych infrastruktur i ich klasyfikacja są trudnym zadaniem ze względu na tę różnorodność i brak spójności nazewniczej oraz technologicznej. Na potrzeby niniejszej pracy analizowano możliwość zaklasyfikowania danej infrastruktury jako GCI biorąc pod uwagę trzy główne aspekty :

1. Algorytmy

   Przyjęto, że do analizy będą brane te rozwiązania, w których zastosowano oprogramowanie autorskie bądź odpowiednio skonfigurowane rozwiązania *open source*. W przypadku gdy platforma oferuje jedynie możliwość użycia zewnętrznego oprogramowania poprzez standardową instalację na platformie, cecha ta nie jest spełniona.

2. Moce obliczeniowe

   Przyjęto, że do analizy będą brane te rozwiązania, w których platforma oferuje zasoby (wątki procesora, przestrzeń dyskowa, pamięć operacyjna, jednostki GPU) do wykorzystania przez jej użytkowników niezależnie od sposobu udostępniania tych zasobów (w postaci maszyn wirtualnych, środowisk programistycznych czy też dynamicznego serwowania oprogramowania podmiotów trzecich).

3. Dane

   Przyjęto, że do analizy będą brane te rozwiązania, w których platforma oferuje dostęp do usystematyzowanego i kompleksowego repozytorium danych przestrzennych. Większość platform oferuje dostęp do zbiorów przykładowych, udostępnionych na potrzeby zapoznania się z ich możliwościami, ale tylko niektóre z nich oferują dostęp do szerokich zbiorów danych przestrzennych.

Na potrzeby analizy porównawczej zaplanowanej w ramach niniejszej rozprawy doktorskiej wzięto pod uwagę przede wszystkim te rozwiązania, które spełniają założenia GCI przynajmniej w zakresie mocy obliczeniowych. Wyniki przeprowadzonej analizy przedstawiono w tabeli 888.

Nieodłącznym elementem definiującym GCI jest zagadnienie wydajnej analizy dużych zbiorów danych przestrzennych. Przeprowadzona analiza wskazuje na dużą różnorodność oferowanych możliwości i zastosowanych narzędzi. Zarówno komercyjne jak i badawcze platformy wykorzystują narzędzia *open source* do przetwarzania rozproszonego. Przykładem mogą być DataBricks[21] oraz CENAGIS[22] (wykorzystujące Apache Spark). Stosowane są również rozwiązania autorskie, takie jak w infrastrukturze CyberGIS[23], jak i zupełnie zamknięte pakiety oprogramowania, izolujące użytkownika od bezpośredniej implementacji poszczególnych algorytmów jak Google Earth Engine[24] czy SentinelHub[25]).

Część platform oferuje też dostęp do repozytorium danych. Platformy GEE oraz SentinelHub udostępniają przede wszystkim dane satelitarne. Platforma CENAGIS udostępnia różnorodne zbiory danych przestrzennych (wektorowe, rastrowe, chmury punktów), jednak w większości

[21] https://www.databricks.com (data dostępu 15.12.2025)
[22] https://cenagis.edu.pl (data dostępu 15.12.2025)
[23] https://cybergis.illinois.edu (data dostępu 15.12.2025)
[24] https://earthengine.google.com (data dostępu 15.12.2025)
[25] https://www.sentinel-hub.com (data dostępu 15.12.2025)

ograniczone do terenu Polski. Amazon CARTO[26] udostępnia usługę Spatial Data Catalog, agregującą publiczne dane przestrzenne oraz zbiory dostępne na zasadach komercyjnych. Pozostałe platformy nie oferują kompleksowego repozytorium danych przestrzennych, a jedynie biblioteki danych przykładowych przeznaczonych głównie do celów edukacyjnych.

Większość z cyberinfrastruktur zapewnia też narzędzia do tworzenia oprogramowania. Wydzielić tutaj można narzędzia z interfejsem graficznym, nie wymagające użycia kodu (np. CARTO Builder[27]) oraz te udostępniające środowisko programistyczne, najczęściej w postaci usługi Jupyter Lab/Jupyter Notebook[28] dostępnej z poziomu przeglądarki internetowej.

Do oddzielnej grupy GCI zaliczyć można klastry badawcze takie jak FASRC[29] (ang. Faculty of Arts and Sciences) z Uniwersytetu Harvarda czy NERC[30] (ang. New England Research Cloud) z Uniwersytetu Nowej Anglii. Nie zostały one stworzone od podstaw jako GCI, ale z czasem zostały wyposażone w oprogramowanie zdolne do wykonywania analiz przestrzennych na dużych zbiorach danych oraz inne elementy cyberinfrastruktury danych przestrzennych.

Wśród istniejących GCI zauważyć można kilka cechujących się wysokim stopniem spełnienia założeń tego typu infrastruktury według (Yang, Raskin, Goodchild i Gahegan, 2010). Są to z pewnością Google Earth Engine (wspomniany nawet w powyższej pracy), Sentinel Hub, Amazon CARTO, CyberGIS oraz CENAGIS. Dwie pierwsze koncentrują się jednak w dużej mierze na danych satelitarnych, podczas gdy infrastruktury Amazon CARTO, CENAGIS oraz CyberGIS skupiają się na wszystkich rodzajach danych przestrzennych. Dwie ostatnie platformy jako jedyne z analizowanych stosują określenia GCI do zdefiniowania zakresu swoich działań.

Zestawienie przedstawione w tabeli 8 nie obejmuje wszystkich istniejących rozwiązań informatycznych przejawiających cechy GCI, gdyż mogą być one odnalezione w większości usług chmurowych (m.in. Microsoft Azure[31] czy Amazon Web Services[32]) czy klastrów badawczych, ale pokazuje kluczowe z nich, ilustrując obecny stan techniki w tym zakresie.

[26] https://carto.com (data dostępu 15.12.2025)
[27] https://carto.com/blog/welcome-to-carto-builder (data dostępu 15.12.2025)
[28] https://jupyter.org (data dostępu 15.12.2025)
[29] https://www.rc.fas.harvard.edu (data dostępu 15.12.2025)
[30] https://nerc.mghpcc.org (data dostępu 15.12.2025)
[31] https://azure.microsoft.com (data dostępu 15.12.2025)
[32] https://aws.amazon.com (data dostępu 15.12.2025)

Tabela 8. Zestawienie istniejących na rynku cyberinfrastruktur danych przestrzennych z rozróżnieniem oferowanych możliwości oraz zastosowanych rozwiązań technologicznych (opracowanie własne)

Nazwa	Google Earth Engine	DataBricks	Amazon CARTO	Sentinel Hub	CyberGIS	CENAGIS	FASRC	NERC
Rodzaj zastosowań	Komercyjny	Komercyjny	Komercyjny	Komerycyjny	Badawczy	Badawczy i Komercyjny	Badawczy	Badawczy
Algorytmy geoprzestrzenne	Tak	Tak	Tak	Tak	Tak	Tak	Nie	Nie
Moce obliczeniowe	Tak	Tak	Tak	Tak	Tak	Tak	Tak	Tak
Dane geoprzestrzenne	Tak	Nie	Tak	Tak	Nie	Tak	Nie	Nie
Rodzaj oprogramowania	autorskie, zamknięte	*open source*	autorskie, zamknięte	autorskie, zamknięte	częściowo autorskie, *open source*	częściowo autorskie, *open source* oraz komercyjne	*open source* i komercyjne	*open source* i komercyjne
Główne rozwiązania technologiczne	Earth Engine API, Earth Engine Code Editor	Apache Spark + Sedona, GeoMesa, RasterFrames, GeoTrellis, Jupyter Notebook i inne	CARTO Builder	SentinelHub API, Jupyter Notebook	CyberGIS-Compute, Jupyter Notebook	Apache Spark + GeoMesa, RasterFrames, GeoTrellis, ArcGIS Enterprise, PostGIS, Jupyter Notebook, Biblioteki CENAGIS i CENAGIS.SAND,	Heavy.ai, HeavyIQ, PostGIS, Jupyter Notebook	ArcGIS Enterprise, KNIME, PostGIS, Heavy.ai

6. Analiza potrzeb i wymagań w kontekście krajowym

Analiza literatury nie wykazała publikacji naukowych odnoszących się do zastosowań metod SBD w analizach przestrzennych dotyczących obszaru Polski, ani prac dla innych terenów polskiego autorstwa w tym zakresie. Nie wyklucza to jednak, że takie badania są prowadzone, tylko wykorzystywane narzędzia SBD mogą nie być explicite wskazywane w opisach metodologicznych. W przeciwieństwie do tego, w przypadku klasycznych technologii GIS dostępna jest obszerna literatura obejmująca zarówno analizy dotyczące Polski, jak i publikacje autorów krajowych. Sugeruje to, że zastosowania SBD w tym kontekście pozostają w literaturze niedostatecznie udokumentowane lub słabiej eksponowane.

Nieco inaczej sytuacja wygląda w kontekście projektów innych niż naukowe. Wykonanie niektórych produktów komercyjnych lub instytucjonalnych mogło wymagać zastosowania metod SBD w skali kraju, a z całą pewnością wykorzystano dane, które spełniają definicję danych SBD. Do przykładów takich produktów zaliczyć można Krajową Mapę Koron Drzew[33], Informatyczny System Obrony Kraju (ISOK) (Kurczyński i Bakuła, 2013) czy ciągłą wizualizację chmury punktów w skali kraju w postaci Cesium 3DTiles[34] utworzoną w ramach prac nad infrastrukturą CENAGIS (Wendland, Gotlib, Choromański, Kaczałek i Stawowski, 2024).

Istniejące źródła nie pozwalają jednak na określenie realnej przydatności, zapotrzebowania i sposobu wykorzystania narzędzi SBD wśród ekspertów GIS. W celu lepszego poznania tych aspektów przeprowadzono ankietę wśród specjalistów zajmujących się analizą danych przestrzennych zarówno w sektorze komercyjnym, naukowym jak i instytucjonalnym. Poniżej zaprezentowano metodykę oraz wyniki przeprowadzonych badań.

6.1. Metodyka badania ankietowego

Przeprowadzona analiza literatury oraz doświadczenia autora w zakresie pracy ze specjalistami GIS w projektach naukowych, komercyjnych, podczas prowadzenia różnego rodzaju szkoleń

[33] https://aplikacja.mapadrzew.com (data dostępu 15.12.2025)
[34] https://github.com/CesiumGS/3d-tiles (data dostępu 15.12.2025)

oraz działalności w roli koordynatora technicznego Politechniki Warszawskiej w ramach Sieci Naukowych Analiz Geoprzestrzennych doprowadziły do założenia następującej hipotezy badawczej dla badania ankietowego: „Eksperci analizy danych przestrzennych w Polsce napotykają i rozwiązują problemy typowe dla danych SBD bez bezpośredniego wykorzystania narzędzi SBD".

W celu sprawdzenia powyższej hipotezy badanie przeprowadzono wśród specjalistów z dziedziny GIS oraz analiz geoprzestrzennych reprezentujących firmy, instytucje publiczne oraz uczelnie. Dobór respondentów oparto o metodę doboru celowego. Do badania zaproszono osoby o znacznym doświadczeniu w przetwarzaniu danych przestrzennych. Celem nie było uogólnienie wyników na całą populację, lecz pogłębiona analiza opinii i doświadczeń specjalistów, ze szczególnym uwzględnieniem zagadnienia przetwarzania dużych zbiorów danych przestrzennych.

Ankietę podzielono na trzy części:

1. Charakterystyka respondentów (wiek, doświadczenie, reprezentowana jednostka, profil zawodowy oraz dziedzina nauki)
2. Stosowane narzędzia i metody analizy GIS (wykorzystywane najczęściej narzędzia, posiadanie umiejętności programistycznych, wykorzystywany sprzęt komputerowy oraz rodzaj przeprowadzanych analiz i wykorzystywanych danych)
3. Problemy i sposób realizacji analiz dużych zbiorów danych (napotykane problemy oraz ich dotkliwość, relacja wielkości zbioru do problemów, metody rozwiązywania problemów oraz zakres analizowanych danych przestrzennych)

Ankietę przeprowadzono w formie on-line z użyciem platformy Microsoft Forms[35]. Udostępniono ją członkom Sieci Naukowej Analiz Geoprzestrzennych oraz pracownikom kluczowych firm i instytucji zajmujących się analizą GIS w Polsce. Dane zebrano w sposób anonimowy, z możliwością podania adresu e-mail jedynie w celach udostępnienia wniosków z ankiety dla zainteresowanych nimi uczestników.

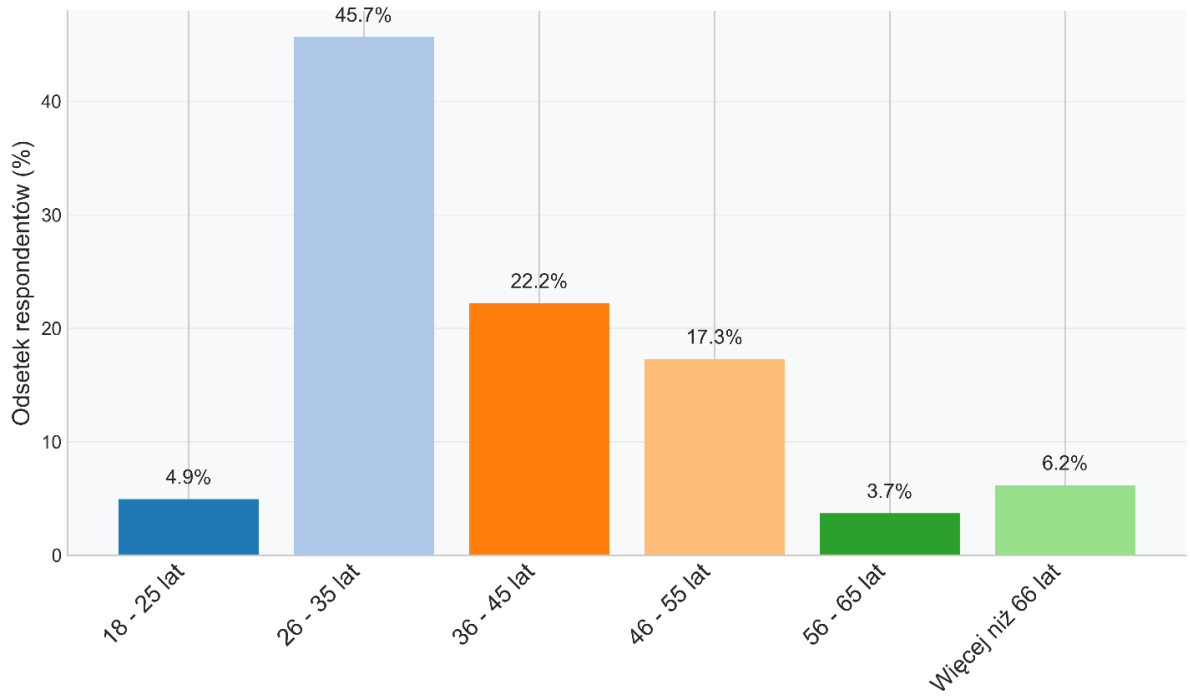
[35] https://forms.office.com (data dostępu 15.12.2025)

6.2. Wyniki badania ankietowego

Wyniki badania ankietowego opracowano i przeanalizowano w podziale na trzy opisane powyżej części. Dane pobrano w oryginalnej formie z platformy Microsoft Forms. Następnie zbiór ten oczyszczono (m.in. agregując odpowiedzi z pytań otwartych) oraz przetworzono do formy wykresów, agregacji i zestawień odpowiedzi na poszczególne pytania.

6.2.1. Charakterystyka respondentów

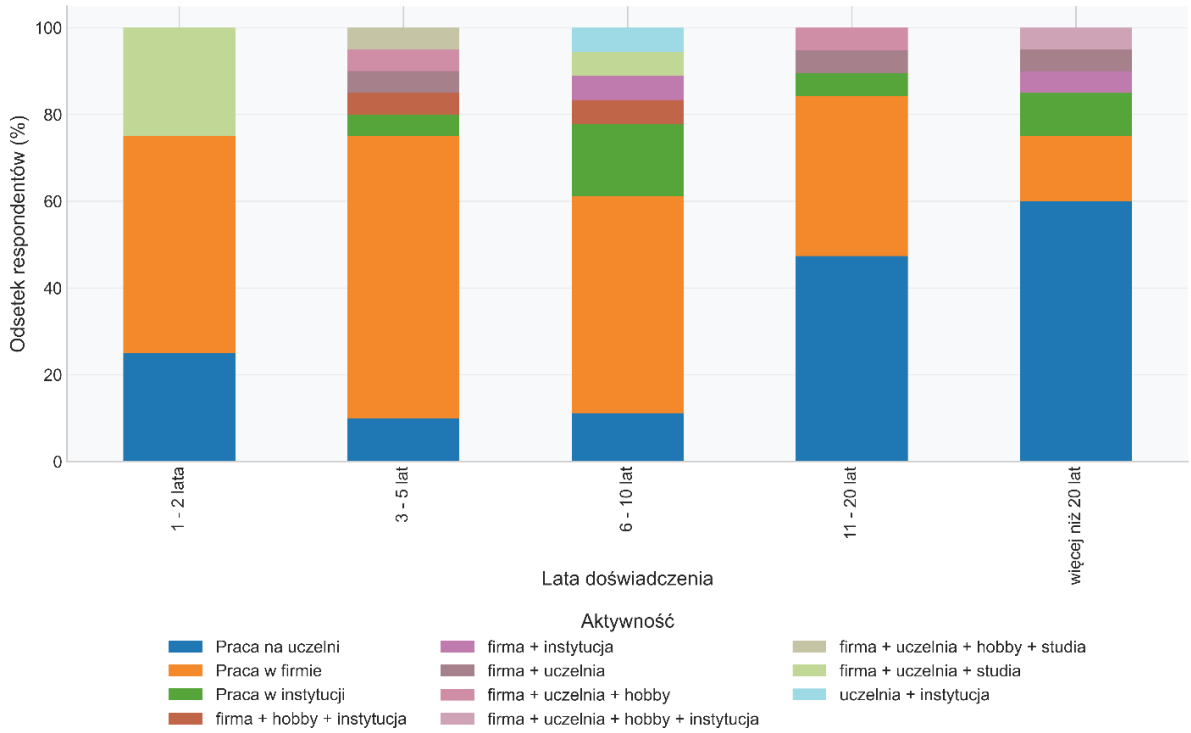
W badaniu ankietowym wzięło udział 81 uczestników z całej Polski. Na rysunku 14 przedstawiono rozkład respondentów według wieku.



Rysunek 14. Rozkład respondentów badań ankietowych według wieku (opracowanie własne)

Zauważyć można, że największa liczba uczestników ankiety znalazła się w przedziale wiekowym 26 – 35 lat. Licznie reprezentowane są też przedziały 36 – 45 lat oraz 46 – 55 lat. Zdecydowana większość respondentów charakteryzowała się też znacznym doświadczeniem zawodowym w analizie danych przestrzennych: Blisko połowa (39 uczestników) zadeklarowała doświadczenie zawodowe powyżej 11 lat, z czego 20 respondentów wskazało, iż posiadają więcej niż 20 lat doświadczenia. Doświadczenie pomiędzy 6, a 10 lat wskazało 18 uczestników, a pomiędzy 3, a 5 lat 20 odpowiadających. Jedynie 4 odpowiedzi zawierały informacje o doświadczeniu mniejszym niż 3 lata.

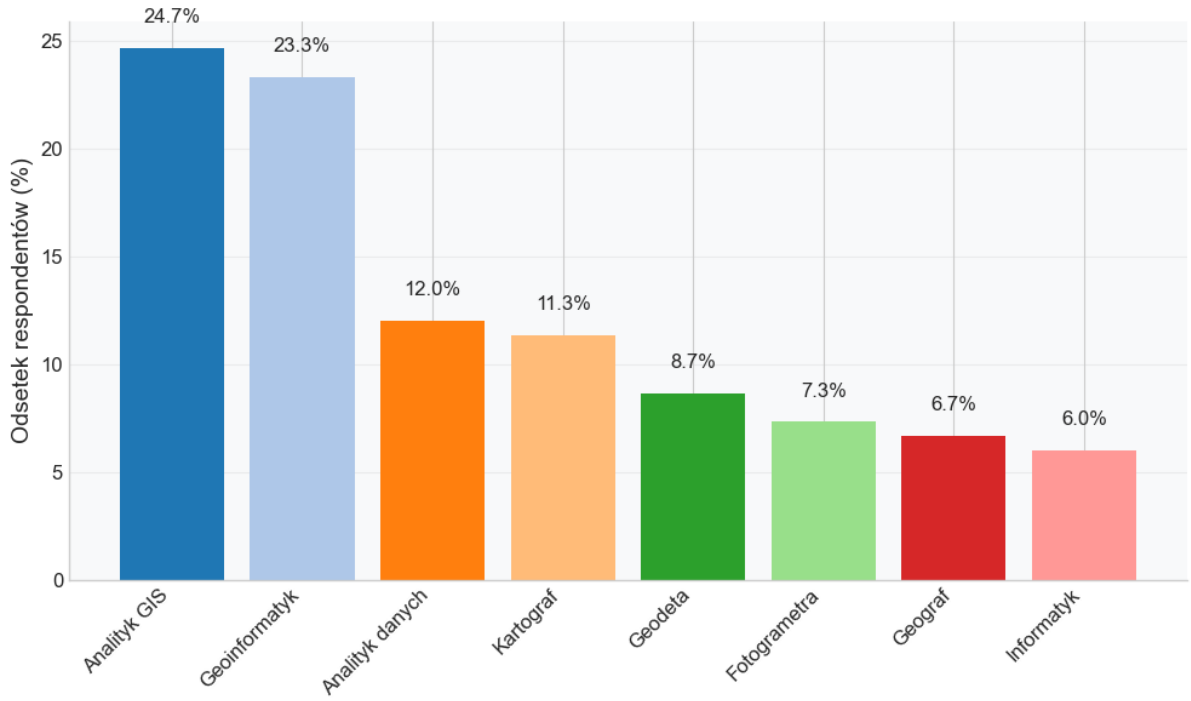
Najwięcej odpowiedzi pochodziło z uczelni (34). Kolejno odpowiedzi pochodziły z firm (26) oraz instytucji. 14 odpowiadających zadeklarowało, że analizy przestrzenne prowadzą w ramach pracy w różnych typach jednostek. Na rysunku 15 przedstawiono zależność rodzaju realizowanych aktywności zawodowych od poziomu doświadczenia.



Rysunek 15. Rozkład realizowanej aktywności zawodowej w zależności od doświadczenia (opracowanie własne)

Analizując ten rozkład zauważyć można, że wraz ze wzrostem stażu w przetwarzaniu danych przestrzennych więcej respondentów podejmuje pracę na uczelni kosztem pracy w firmie. Żaden z odpowiadających nie wskazał też hobby jako jedyny powód wykonywania analiz przestrzennych.

Na rysunku 16 przedstawiono rozkład respondentów w zależności od reprezentowanej profesji.



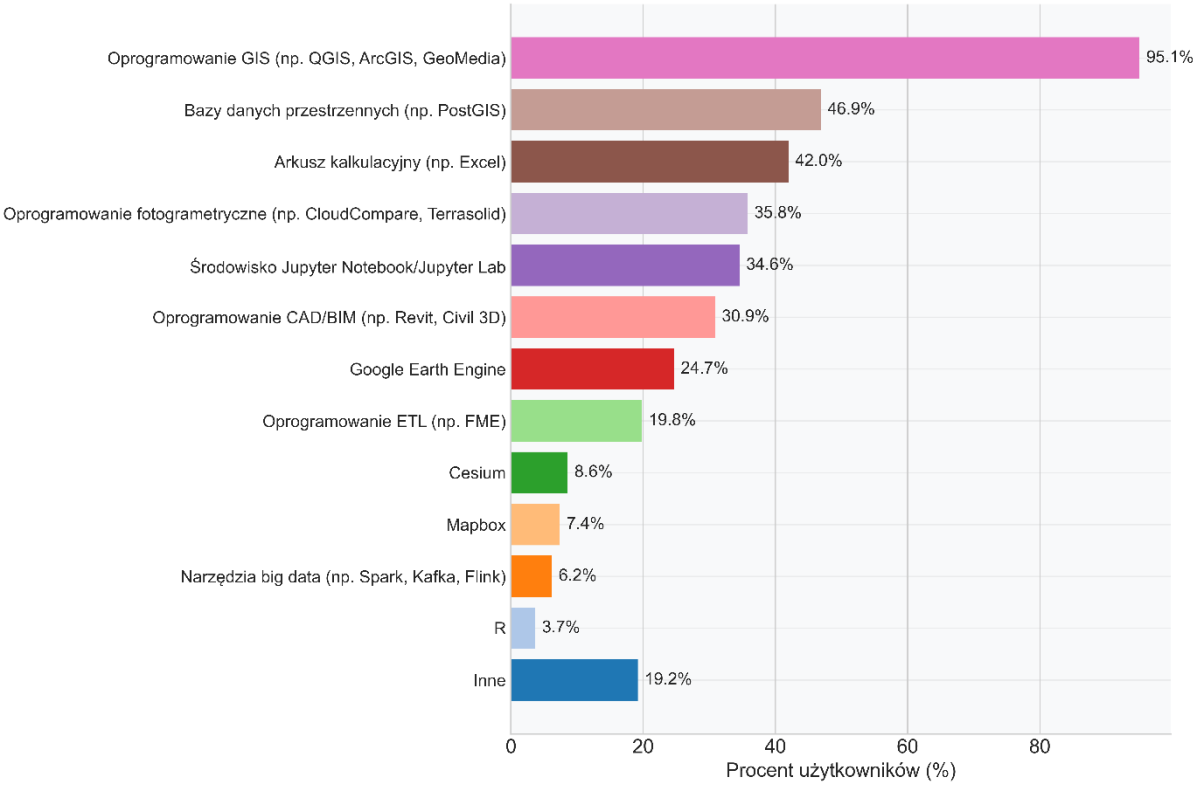
Rysunek 16. Rozkład respondentów według zadeklarowanej profilu zawodowego (opracowanie własne)

Do dwóch najliczniejszych grup należeli analitycy GIS oraz geoinformatycy. W dalszej kolejności licznie reprezentowane były zawody analityka danych oraz kartografa. Zdecydowana większość respondentów reprezentowała nauki inżynieryjno-techniczne (56) lub ścisłe i przyrodnicze (16).

Analiza charakterystyki respondentów pozwala stwierdzić, iż ankieta trafiła do zakładanej grupy docelowej specjalistów analizy danych przestrzennych o wysokim doświadczeniu zawodowym. Odpowiadający reprezentują trzy podstawowe sektory działalności (jednostki naukowe, firmy, instytucje), a także w dość równomiernym stopniu obrazują doświadczenia osób o różnym wieku oraz deklarowanej profesji.

6.2.2. Stosowane narzędzia i metody analizy GIS

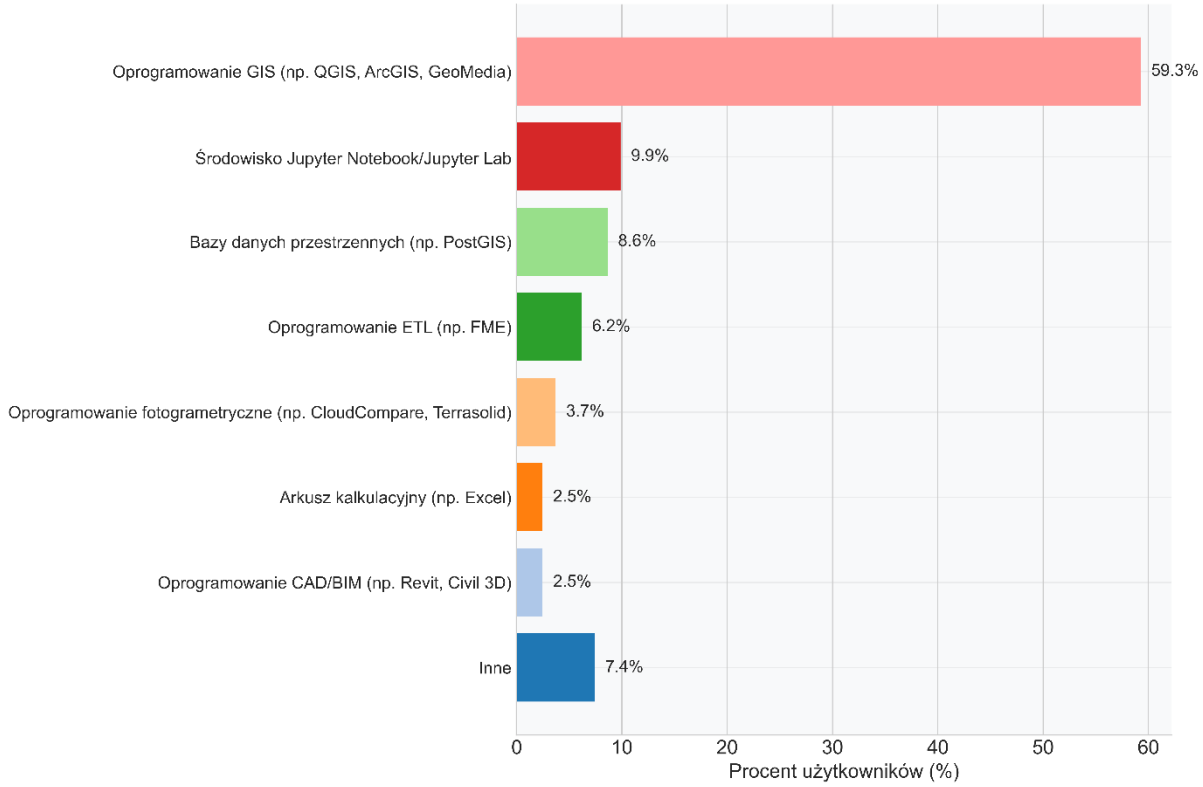
W odpowiedzi na pytanie wielokrotnego wyboru „Jakich narzędzi używasz do analizy danych przestrzennych?” aż 95.1% ankietowanych wskazało na klasyczne oprogramowanie desktop GIS. Dużą popularnością cieszyły się też systemy baz danych przestrzennych oraz arkusz kalkulacyjny. 6.2% respondentów wskazało w tym pytaniu na specjalistyczne narzędzia SBD. Na rysunku 17 przedstawiono szczegółowe zestawienie odpowiedzi.



Rysunek 17. Zestawienie narzędzi wykorzystywanych do analiz danych przestrzennych (opracowanie własne)

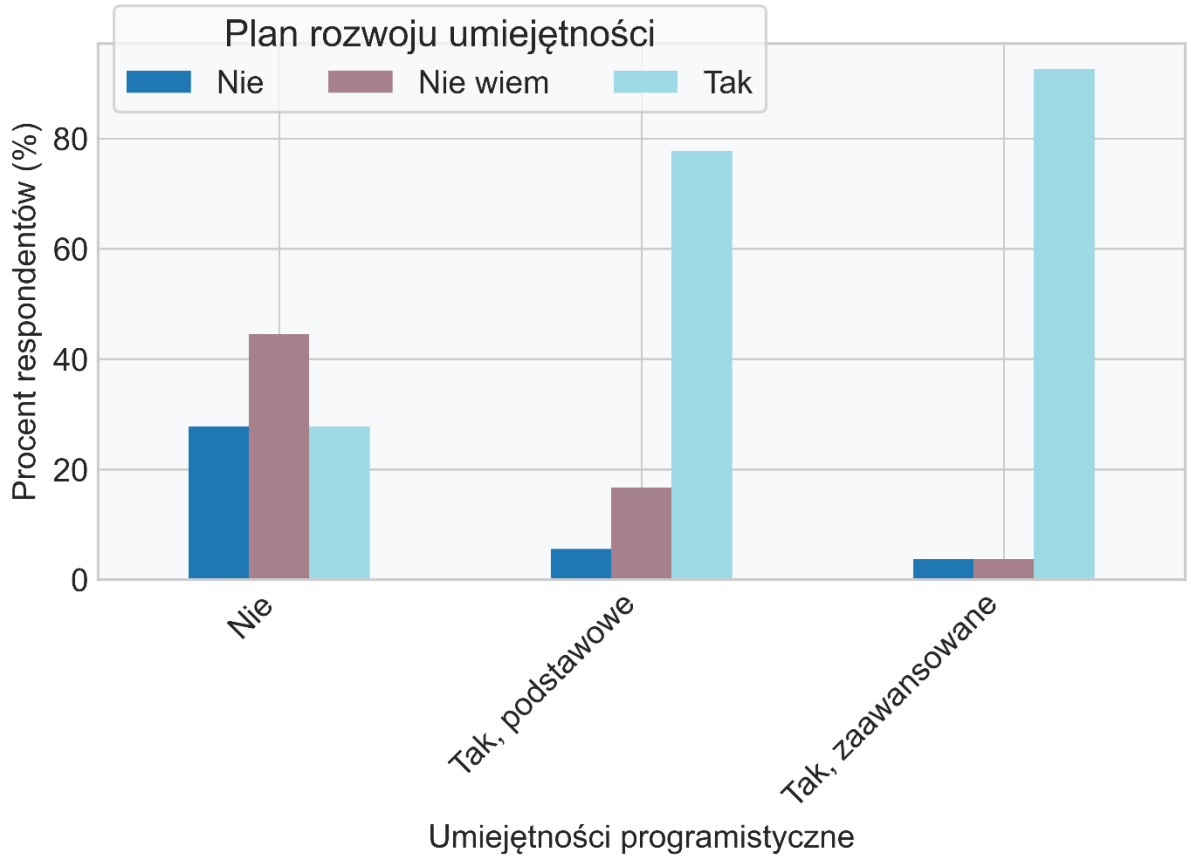
Selekcja wykorzystywanych narzędzi właściwie nie zmienia się w zależności od doświadczenia. We wszystkich grupach pod względem lat doświadczenia zawodowego popularne są te same narzędzia (oprogramowanie GIS, bazy danych przestrzennych, arkusz kalkulacyjny, oprogramowanie fotogrametryczne oraz środowisko Jupyter).

Powyższe pytanie zostało pogłębione przez pytanie jednokrotnego wyboru o treści „Jakiego narzędzia używasz najczęściej do analizy danych przestrzennych?". Pozwoliło one na wskazanie głównych narzędzi pracy respondentów. W tym przypadku blisko 60% wskazało oprogramowanie desktop GIS, a niemal 10% wskazało środowisko Jupyter jako narzędzie pierwszego wyboru. Żaden z respondentów nie wskazał w tym pytaniu narzędzi SBD. Szczegółowy rozkład odpowiedzi na to pytanie zaprezentowano na rysunku 18.



Rysunek 18. Narzędzie pierwszego wyboru używane do analiz danych przestrzennych (opracowanie własne)

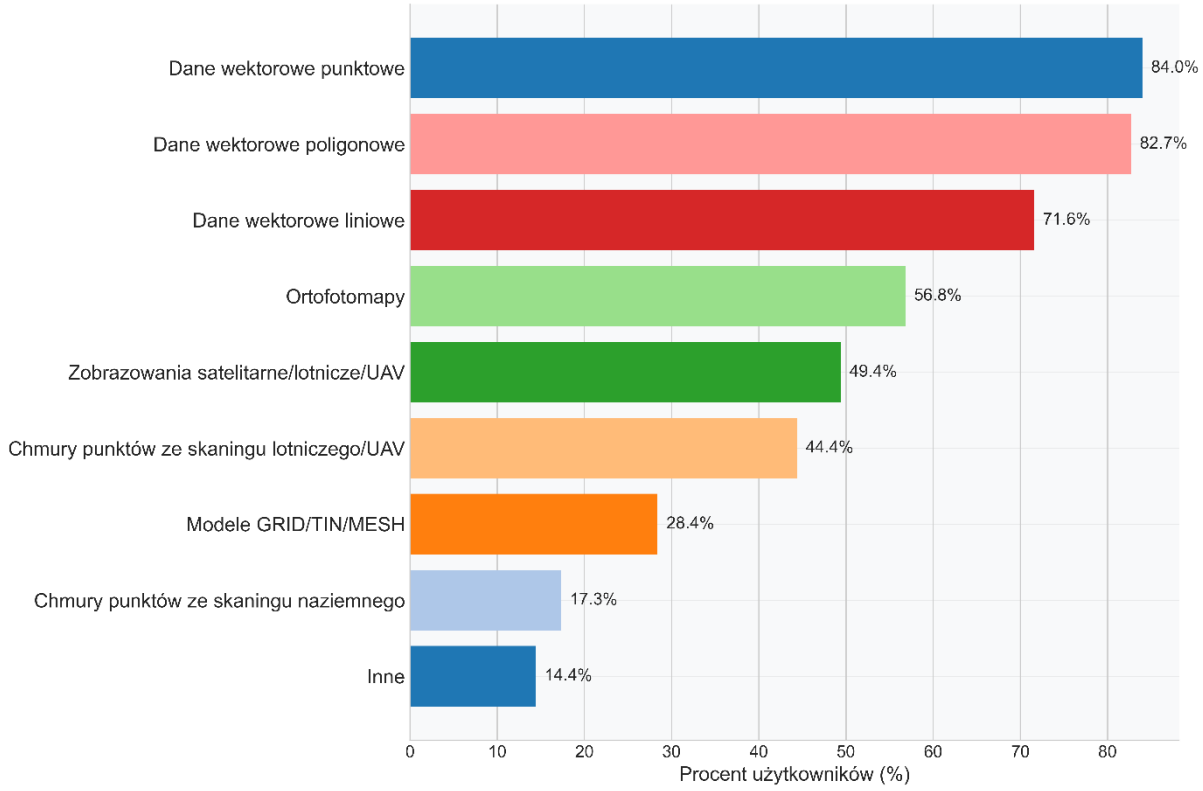
Kolejnym istotnym aspektem pracy specjalisty GIS, szczególnie w kontekście analiz dużych zbiorów danych przestrzennych, jest programowanie. 78% respondentów zadeklarowało posiadanie umiejętności programistycznych, z czego 33% określiło poziom swoich umiejętności jako zaawansowany. W pytaniu dotyczącym planów dalszego rozwoju umiejętności (rysunek 19) zauważyć można wyraźną chęć rozszerzenia wiedzy z zakresu programowania przez tych uczestników badania, którzy już posiadają przynajmniej podstawowe umiejętności programistyczne. Najpopularniejszym językiem programowania wykorzystywanym w analiza przestrzennych jest język Python (55.3% respondentów). W dalszej kolejności stosowane są jeżyki JavaScript oraz R (oba po 12.6%), a 5.3% badanych deklaruje, że nie używa języków programowania do przeprowadzania analiz przestrzennych pomimo posiadania umiejętności programistycznych. Wśród innych języków programowania pojawiają się języki takie jak C, Visual Basic, C# czy Kotlin.



Rysunek 19. Plany rozwoju umiejętności programistycznych w zależności od obecnie posiadanych umiejętności (opracowanie własne)

Kolejna sekcja drugiej części ankiety poświęcona była wykorzystywanemu sprzętowi komputerowemu oraz analizowanym danym. Ponad 90% odpowiadających korzysta z komputerów desktop bądź laptopów do analizy danych. Stosunkowo często jednak wykorzystywane są też technologie chmurowe (blisko 40% respondentów), a dedykowane stacje robocze wykorzystywane są przez blisko 30% respondentów.

Pytanie wielokrotnego wyboru „Jakie rodzaje danych przestrzennych wykorzystujesz najczęściej w analizach?" miało na celu określenie jakie typy danych cieszą się największą popularnością (przydatnością) wśród specjalistów GIS. Rozkład odpowiedzi na to pytanie przedstawiono na rysunku 20. Na pierwszym miejscu znalazły się dane wektorowe punktowe (84%), poligonowe (83%) oraz liniowe (72%). W dalszej kolejności odpowiadający wykorzystują dane rastrowe, takie jak ortofotomapy czy zobrazowania satelitarne, lotnicze bądź pochodzące z bezzałogowych statków powietrznych.



Rysunek 20. Rozkład odpowiedzi na pytanie „Jakie rodzaje danych przestrzennych wykorzystujesz najczęściej w analizach?" (opracowanie własne)

Analiza powiązań pomiędzy odpowiedziami wskazuje, że popularność danych wektorowych w odpowiedziach wynika m. in. z tego, że są one bardzo często używane wspólnie z innymi zbiorami danych, w szczególności z danymi rastrowymi. Nie występują znaczne różnice pomiędzy wykorzystywanymi danymi, a preferowanymi kategoriami sprzętu. Zauważyć można jedynie lekką, względną przewagę danych rastrowych i chmur punktów nad wektorami w przypadku dedykowanych stacji roboczych, co może być spodziewane ze względu na charakterystykę przetwarzania tych rodzajów danych (konieczność dostępu do licencji, wykorzystanie wysokowydajnych jednostek GPU oraz przetwarzania wielowątkowego).

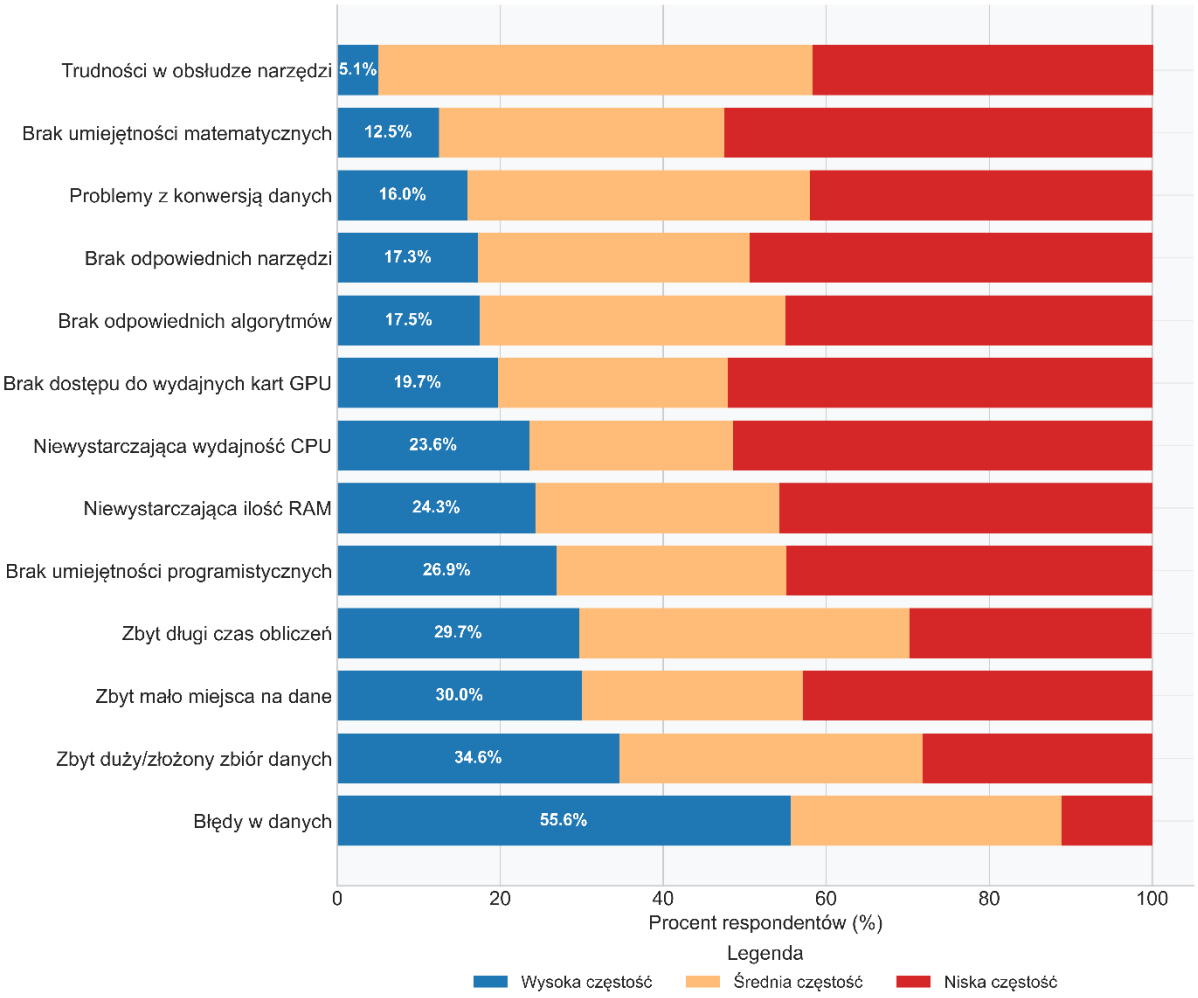
Uczestników ankiety poproszono też o podanie konkretnych zagadnień, którymi zajmują się w kontekście analizy danych przestrzennych, co pozwoliło na określenie głównych kierunków rozwoju i zainteresowań analityków GIS. Na rysunku 21 zaprezentowano chmurę słów najczęściej występujących w odpowiedzi na to pytanie otwarte. Respondenci wskazali szeroki zakres zagadnień analizy danych przestrzennych – od zagadnień geostatystycznych, przez fotogrametryczne i związane z analizą obrazów aż do modelowania BIM i zastosowań sztucznej inteligencji.

Rysunek 21. Chmura słów utworzona na podstawie odpowiedzi ankietowanych na pytanie "Podaj konkretne zagadnienia, którymi zajmujesz się w kontekście analizy danych przestrzennych" (opracowanie własne)

6.2.3. Problemy w analizie dużych zbiorów danych

Pierwsze pytanie z kolejnej części ankiety wymagało od uczestników wskazania częstości występowania każdego z 13. problemów. Ich treść oraz zestawienie odpowiedzi przedstawiono na rysunku 22. Problemem określanym jako występujący najczęściej są błędy w danych. Ponad połowa odpowiadających określiła ten problem jako występujący często, bardzo często lub zawsze. Drugim w kolejności najczęściej występującym problemem jest zbyt duży, bądź bardzo złożony zbiór danych. Kolejno występują problemy potencjalnie skorelowane z wielkością zbioru tj. zbyt długi czas obliczeń oraz zbyt mało miejsca na dane. Niespełna 27% respondentów identyfikuje niedobór umiejętności programistycznych jako źródło problemów w analizie danych przestrzennych. Niewystarczająca wydajność CPU oraz brak wystarczającej ilości pamięci RAM są identyfikowane jako często występujący problem przez zbliżoną liczbę respondentów (odpowiednio 23.6% i 24.3%). Problem sprzętowy z GPU identyfikowany jest rzadziej – u 19.7% odpowiadających, co może być związane z ogólnym rzadszym wykorzystaniem tych jednostek obliczeniowych w analizach przestrzennych. Odpowiedzi wskazują też na wysoki rozwój umiejętności obsługi narzędzi do analizy przestrzennej –

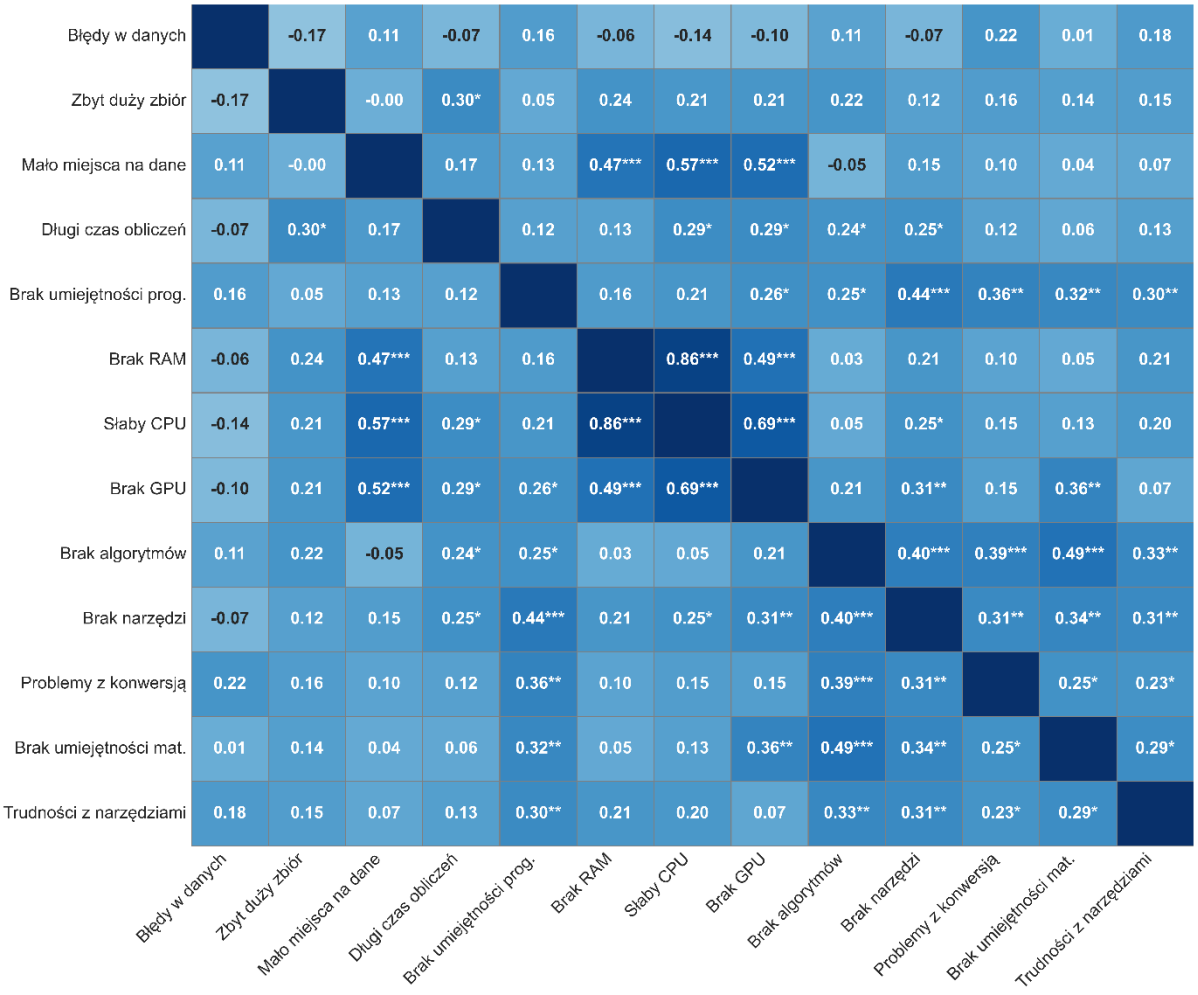
problemy z brakiem odpowiednich narzędzi, algorytmów czy konwersją danych należą do rzadko występujących.



Rysunek 22. Wizualizacja wyników odpowiedzi na pytanie "Wskaż jak często napotykasz na poniższe problemy w trakcie analizy danych przestrzennych" (opracowanie własne)

W celu zbadania współzależności opisanych problemów przeprowadzono analizę korelacji z wykorzystaniem współczynnika Spearmana (Spearman, 1904). Analiza pozwoliła ocenić współwystępowanie poszczególnych problemów, wraz z asymptotyczną p-wartością istotności (zgodnie z implementacją funkcji spearmanr w pakiecie SciPy[36]). Na rysunku 23 zaprezentowano wynikową macierz korelacji.

[36] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html (data dostępu 15.12.2025)



Rysunek 23. Macierz korelacji Spearmana wraz z oznaczeniem p-wartości (* p<0.05, ** p<0.01, *** p<0.001) dla odpowiedzi na pytanie "Wskaż jak często napotykasz na poniższe problemy w trakcie analiz danych przestrzennych" (opracowanie własne)

Analiza korelacji Spearmana pomiędzy zmiennymi wskazuje na kilka interesujących zależności. Silną wzajemną korelacje zauważyć można pomiędzy wszystkimi problemami związanymi z elementami składowymi infrastruktury komputerowej (RAM, CPU, GPU i miejsce na dane), co może świadczyć o częstym występowaniu tych problemów wspólnie. Problemy narzędziowe (brak algorytmów, trudności z narzędziami, brak narzędzi) również charakteryzują się silną korelacją. Z kolei brak umiejętności programistycznych jest powiązany z problemami narzędziowymi. Istnieje zauważalna statystycznie korelacja pomiędzy występowaniem problemów ze zbyt dużym zbiorem zdanych, a długim czasem obliczeń. Wielkość tej korelacji sugeruje jednak, że problemy te występują również rozłącznie. Podobna sytuacja występuje w przypadku powiązania długiego czasu obliczeń z brakami w zakresie wydajności jednostek obliczeniowych (CPU, GPU). Nie zachodzi jednak mocna korelacja pomiędzy stwierdzeniem

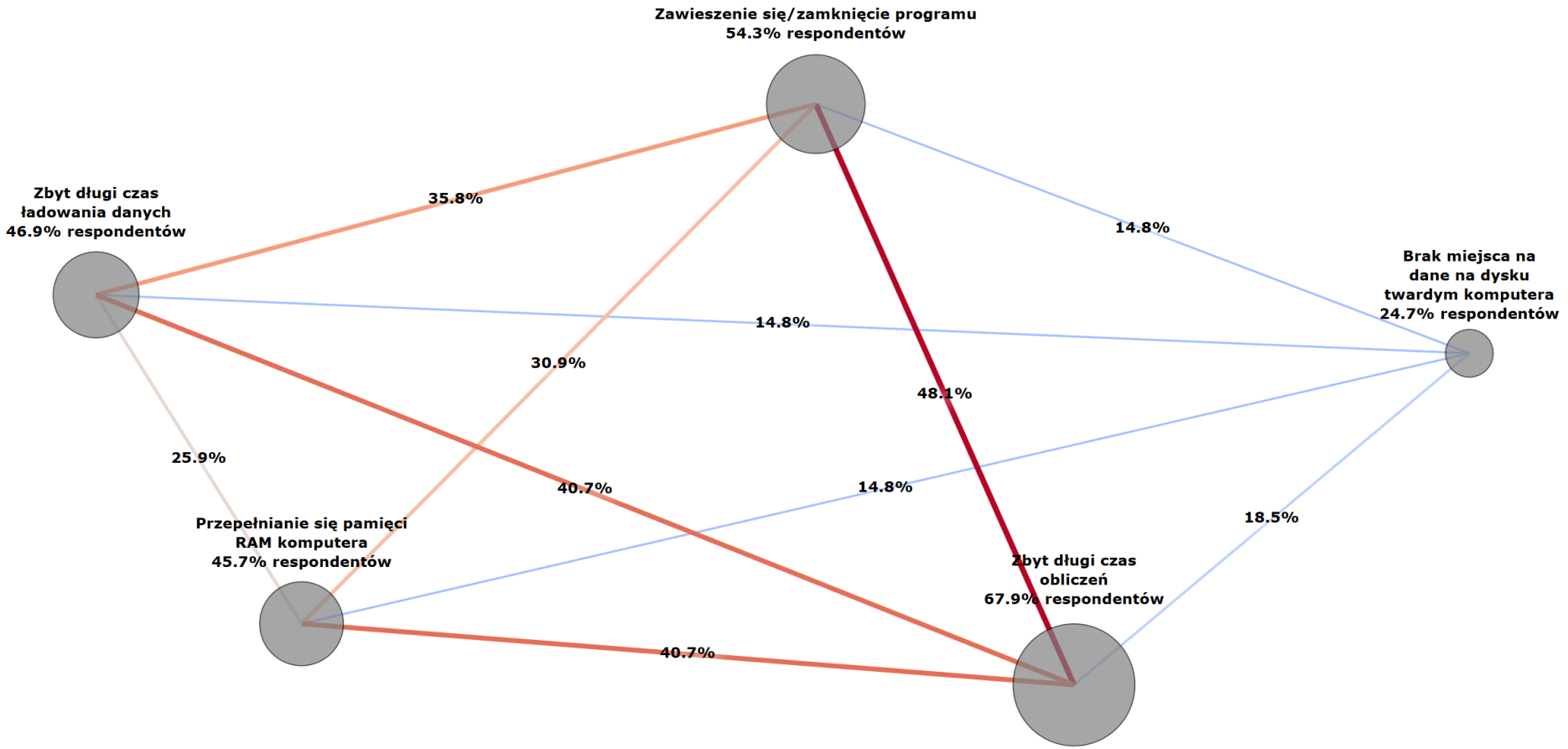
o problemach związanych ze zbyt dużą wielkością/złożonością zbioru danych, a sprzętem czy narzędziami.

Przeanalizowano również dwa najczęściej występujące problemy (błędy w danych oraz zbyt duży zbiór danych) w relacji do doświadczenia zawodowego respondentów. W przypadku błędów w danych wszystkie osoby odpowiedziały stosunkowo podobnie. Problemy związane ze zbyt dużym zbiorem przetwarzanych danych są rzadziej identyfikowane w grupie odpowiadających z największym doświadczeniem zawodowym. W innych grupach jest to ok 80% wszystkich odpowiedzi, a w tej grupie jest to około 50%. Może to wskazywać na przydatność rozległego doświadczenia w rozwiązywaniu tego typu problemów lub większe skupienie na mniejszych zbiorach danych w tej grupie stażu pracy.

Drugie pytanie w tej części ankiety brzmiało „Czy podczas pracy z danymi przestrzennymi napotkałeś/aś na problem związany ze zbyt dużą wielkością i/lub złożonością zbioru danych względem wykorzystywanego narzędzia?". 79% ankietowanych odpowiedziało na to pytanie twierdząco. Łącząc tą informację, z faktem, że problem ten jest częsty lub bardzo częsty dla blisko 35% ankietowanych można stwierdzić, że to zagadnienie stanowi jeden z poważniejszych problemów, przed którym stają specjaliści GIS w Polsce. Respondenci deklarujący problemy z przetwarzaniem dużych zbiorów danych opierają swoje działania na wszystkich rodzajach danych przestrzennych. Zauważyć można jednak minimalnie częstsze występowanie problemów dla respondentów wykorzystujących dane obrazowe i chmury punktów, a rzadsze w przypadku modeli GRID/TIN/MESH. Najrzadziej problemy z przetwarzaniem dużych zbiorów danych deklarowali użytkownicy komputerów desktop, a najczęściej dedykowanych stacji roboczych. Różnice nie są tutaj jednak duże i mogą wynikać ze specyfiki przetwarzanych danych.

Ankietowani, którzy odpowiedzieli twierdząco na pytanie „Czy podczas pracy z danymi przestrzennymi napotkałeś/aś na problem związany ze zbyt dużą wielkością i/lub złożonością zbioru danych względem wykorzystywanego narzędzia?". odpowiadali także na dodatkowe pytania dotyczące źródeł oraz sposobów na rozwiązanie tych problemów. Pierwsze pytanie z tej serii dotyczyło najczęstszych objawów problemów. Na rysunku 24 przedstawiono wyniki odpowiedzi na to pytanie. Najczęstszym objawem tych problemów okazał się zbyt długi czas obliczeń, który występował w wielu przypadkach wspólnie z zawieszaniem/zamykaniem

programu. Respondenci wskazali również problemy związane ze zbyt długim czasem ładowania danych oraz przepełnianiem się pamięci RAM komputera. Najrzadziej wskazywanym problemem był brak miejsca na dysku. Rozkład odpowiedzi na to pytanie można podsumować stwierdzeniem, iż najczęściej problemy dotyczą zasobów wykorzystywanych bezpośrednio do obliczeń lub optymalizacji samych algorytmów, a nie miejsca do składowania dużych zbiorów danych.

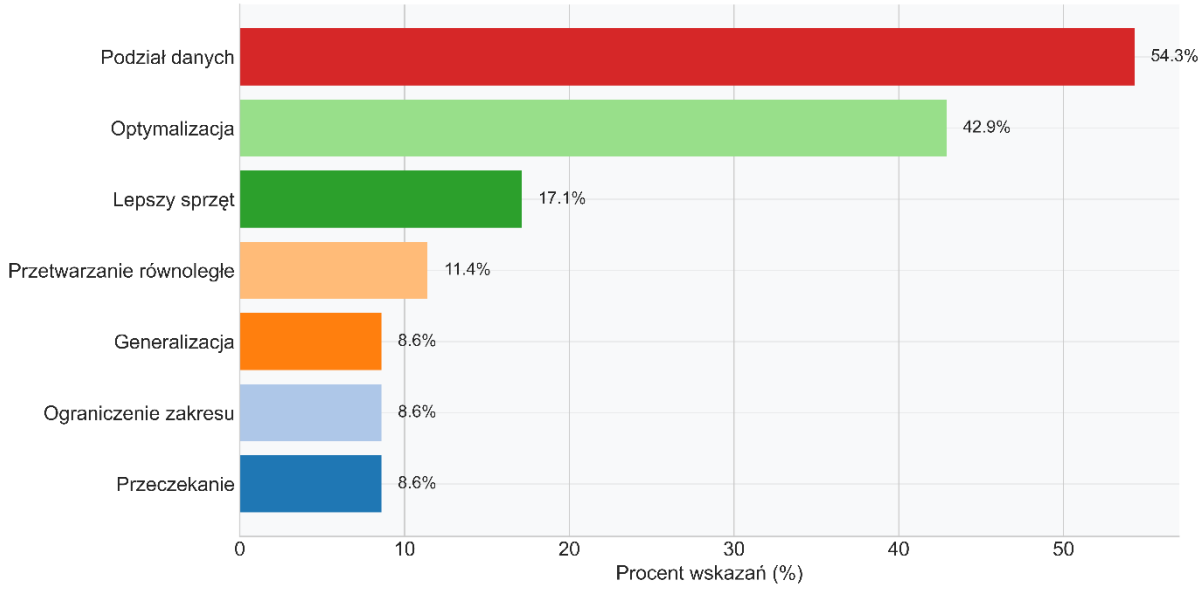


Rysunek 24. Graf utworzony na podstawie odpowiedzi na pytanie „W jaki sposób najczęściej przejawiają się problemy ze zbyt dużą wielkością/złożonością zbioru danych przestrzennych?”. W wierzchołkach grafu zaprezentowano częstość występowania poszczególnych objawów, a na krawędziach oznaczono częstość wspólnego występowania par objawów połączonych daną krawędzią (opracowanie własne)

Następnie, uczestnicy badania zostali poproszeni o odpowiedź na następujące pytanie: „W jaki sposób najczęściej rozwiązujesz problemy ze zbyt dużą wielkością i/lub złożonością zbioru danych?”. Miało ono strukturę otwartą w celu umożliwienia swobodnego wskazania rozwiązań bez narzucania pomysłów w samej treści pytania. W związku z tym podczas analizy wyników ankiety konieczna była agregacja udzielonych odpowiedzi. Wszystkie odpowiedzi w tym pytaniu przydzielono do następujących kategorii:

- Podział danych – wytworzenie podzbiorów, kafli, próbek, partycji oryginalnego zbioru danych, czyli rozbicie większego problemu na kilka mniejszych
- Optymalizacja – wdrożenie optymalizacji algorytmu, indeksowanie przestrzenne, zmiana formatu wejściowego, zmiana oprogramowania na bardziej wydajne
- Lepszy sprzęt – rozszerzenie zasobów obecnie posiadanego sprzętu komputerowego bądź wykorzystanie zewnętrznych serwerów obliczeniowych
- Generalizacja – uproszczenie geometrii, zmniejszenie rozdzielczości danych
- Przetwarzanie równoległe – przygotowanie wielowątkowych wersji algorytmów, uruchomienie obliczeń na wielu komputerach na raz
- Ograniczenie zakresu – obcięcie zakresu przestrzennego danych – wykonanie przetworzenia w ograniczonym niż założony zakres
- Przeczekanie – uruchomienie obliczeń i pozostawienie ich na dłuższy czas (przez noc, weekend, tydzień)

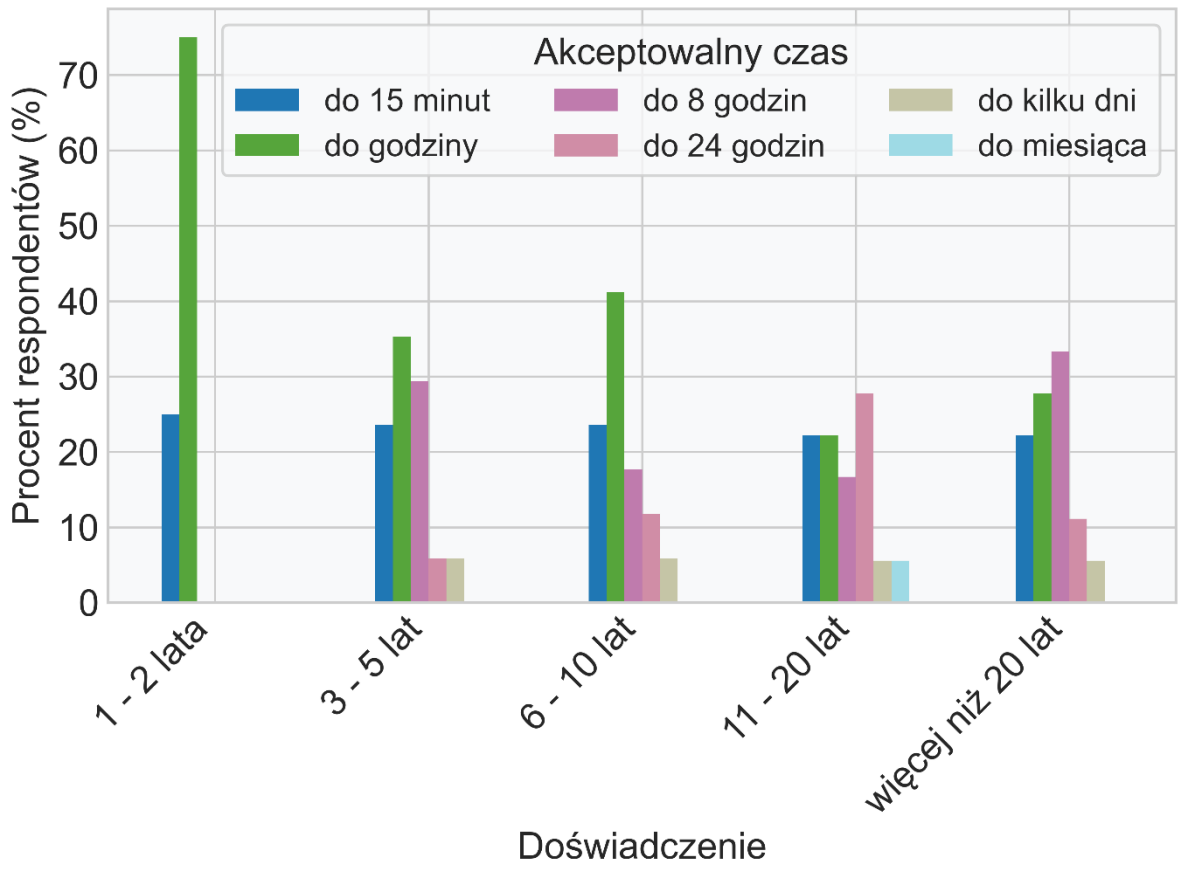
Pytanie to nie było obowiązkowe, odpowiedzi udzieliło 36 z 81 badanych, wielu badanych wskazywało kilka rozwiązań problemów w jednej wypowiedzi – w takim wypadku daną odpowiedź przypisywano do kilku kategorii. Na rysunku 25 zestawiono poszczególne sposoby rozwiązań w kolejności od najczęściej wskazywanego.



Rysunek 25. Zestawienie sposobów rozwiązywania problemów z dużymi zbiorami danych przestrzennych na podstawie odpowiedzi ankietowanych na pytanie „W jaki sposób najczęściej rozwiązujesz problemy ze zbyt dużą wielkością i/lub złożonością zbioru danych?" (opracowanie własne)

Jak wynika z analizy odpowiedzi respondentów, najczęstszym sposobem rozwiązania problemów jest podział danych na mniejsze fragmenty (54.3% odpowiedzi). Następna w kolejności jest optymalizacja używanych algorytmów (42.9%) oraz zastosowanie lepszego sprzętu komputerowego (17.1%). Należy zauważyć, że pomysły oparte o przetwarzanie równoległe wystąpiły jedynie w 11.4% odpowiedzi. Można jednak przypuszczać, że część respondentów założyła takie przetwarzanie mając na myśli podział danych na mniejsze części. Wyraźnie widać, że badani specjaliści analiz GIS nie chcą lub nie mogą (ze względu na cel badania) rozwiązywać problemów poprzez ograniczanie danych wejściowych (generalizacja, ograniczanie zakresu).

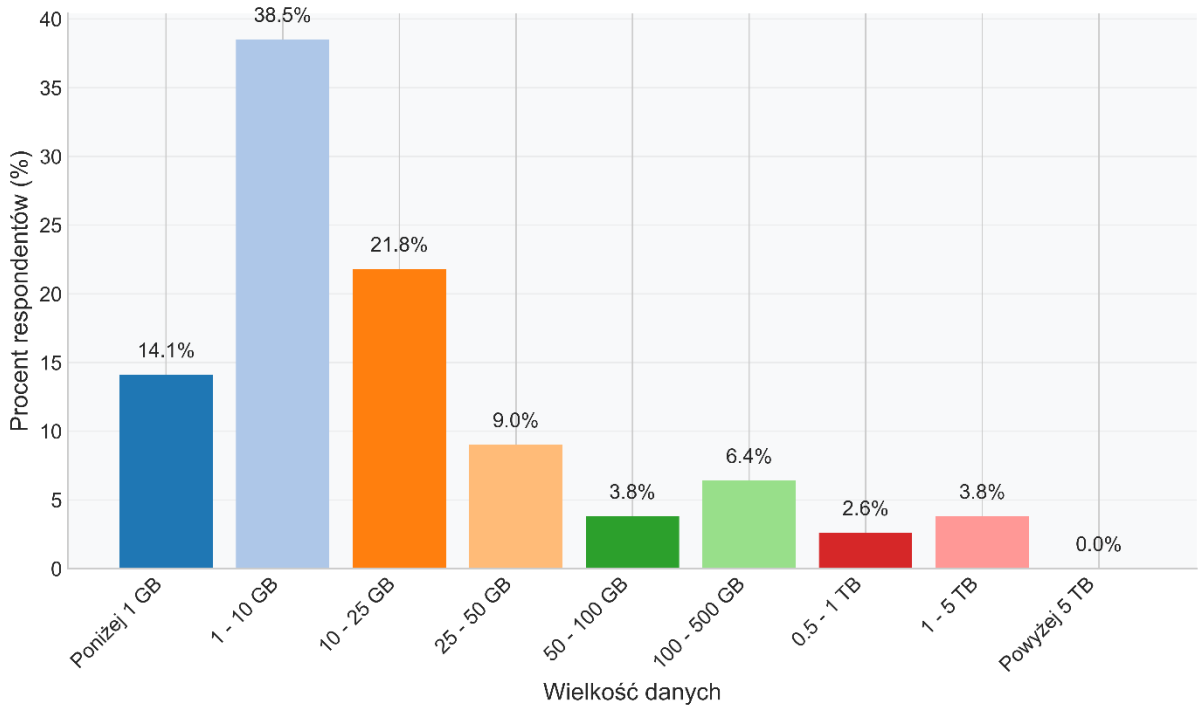
Kolejne pytanie dotyczyło maksymalnego akceptowalnego czasu obliczeń w ramach realizowanych analiz przestrzennych. Większość respondentów określiła, że krótki czas obliczeń (do godziny) jest akceptowalny, z czego ponad ¼ badanych wskazuje, że oczekiwałaby czasu obliczeń poniżej 15 minut. Jedynie 5.1% badanych akceptuje kilkudniowe oczekiwanie na wynik analizy. Jednocześnie należy podkreślić fakt wskazany przez troje respondentów: akceptowalny czas zależny jest od specyfiki konkretnych obliczeń. Wraz ze wzrostem doświadczenia zauważyć można dłuższy akceptowalny czas obliczeń (rysunek 26). Grupą dominującą w przedziałach doświadczenia 1 – 10 lat jest czas do jednej godziny. Zmienia się to w bardziej zaawansowanych grupach, gdzie wyższe wyniki osiągają czasy 8 i 24 godziny.



Rysunek 26. Zestawienie akceptowalnych czasów obliczeń w zależności od doświadczenia zawodowego na podstawie pytania „Jaki czas obliczeń uważasz za akceptowalny podczas przeprowadzania realizowanych przez Ciebie analiz przestrzennych?" (opracowanie własne)

Pytanie o czas obliczeń uzupełniało pytanie o najczęściej przetwarzaną wielkość danych w ramach jednej analizy (rysunek 27). Największa część odpowiadających określa przedział od 1 do 10 GB danych jako najczęściej przetwarzany w ramach jednej analizy. Drugi najczęściej wskazywany zakres danych to 10 – 25 GB. Specjaliści analizujący dane w zakresie 50 GB i większym stanowią łącznie 16.8% odpowiadających. Należy zauważyć, że żaden z odpowiadających nie wybrał odpowiedzi „Powyżej 5 TB". W zasadzie zgodnie z wcześniej podanymi definicjami w rozdziale 4 (tabela 4) można stwierdzić, że nikt nie pracuje na prawdziwych danych typu *big data*. Interesujące obserwacje spowodowało także zestawienie odpowiedzi na to pytanie z pytaniem o występowanie problemów związanych ze zbyt dużym/złożonym zbiorem danych. Wszyscy odpowiadający, którzy wskazali zakres 50GB i większy znajdują się w grupie deklarującej napotkanie problemów z przetwarzaniem zbyt dużych/złożonych zbiorów danych. Można z tego wnioskować, że ten pułap wielkości danych

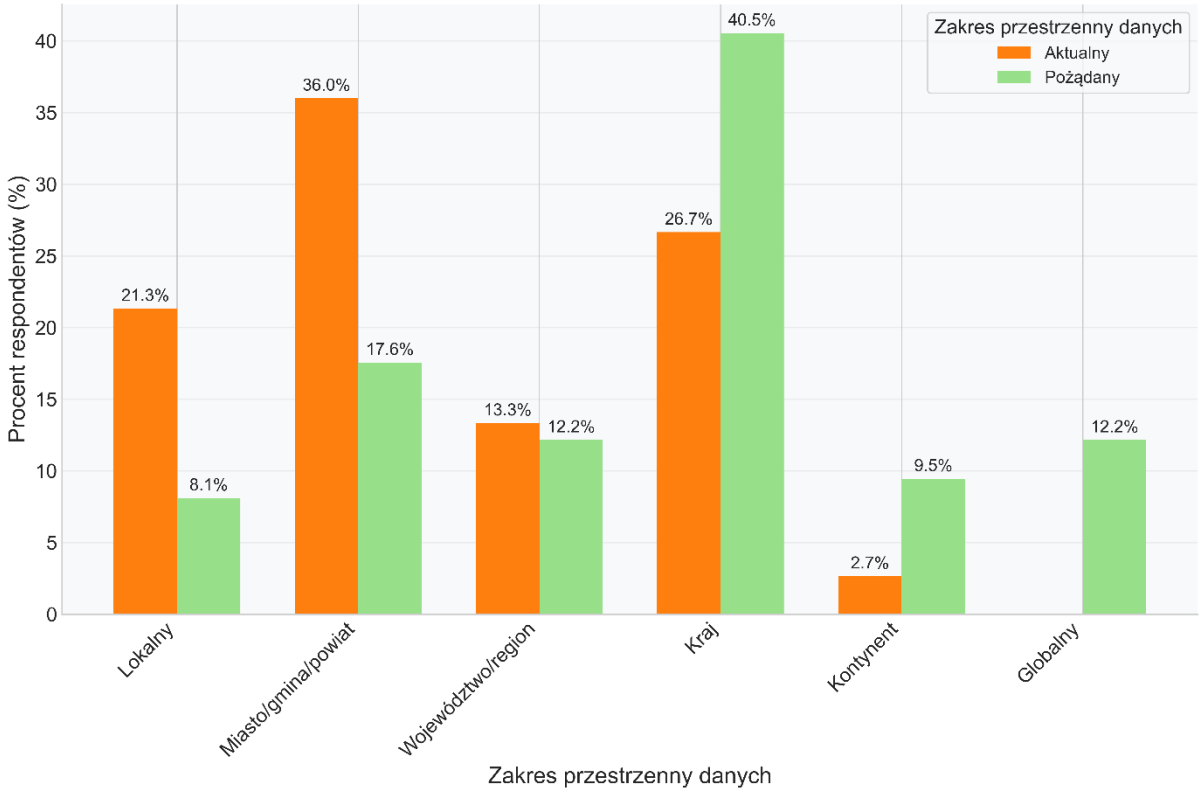
przestrzennych jest już w wielu przypadkach problematyczny i wymaga zastosowania specjalistycznych rozwiązań.



Rysunek 27. Zestawienie wielkości przetwarzanych danych przestrzennych utworzone w oparciu o odpowiedzi na pytanie „Jaką wielkość danych najczęściej przetwarzasz w ramach jednej analizy?" (opracowanie własne)

Ostatnia para pytań dotyczyła aktualnie przetwarzanego oraz oczekiwanego zakresu przestrzennego przetwarzania danych przestrzennych. Na rysunku 28 przedstawiono zestawienie odpowiedzi wskazanych przez ankietowanych na oba te pytania. Odpowiadający przetwarzają najczęściej dane w zakresie miast, gmin oraz powiatów (36%). Popularne zakresy przetwarzania danych to również zakres lokalny (21.3%) oraz zakres całego kraju (26.7%). Jednak w przypadku zakresu pożądanego sytuacja wygląda inaczej. Najbardziej pożądany zakres przestrzenny analizy to zakres całego kraju (40.5%). 12.2% respondentów wskazuje również, że chętnie przetwarzałaby dane w skali całego świata, przy czym realnie żaden z nich teraz danych w takim zakresie nie przetwarza. Większość respondentów zadeklarowała chęć podwyższenia zakresu przestrzennego przetwarzanych danych. Najwięcej z nich (32.9%) oczekiwałaby możliwości przetwarzania o 1 zakres więcej, a 26% o 2 zakresy więcej. Grupa 31.5% nie widzi zysku w możliwości poszerzenia zakresu przestrzennego analizowanych danych. Zestawienia odpowiedzi o zakresie przestrzennym analizowanych danych oraz deklarowanej wielkości tych danych nie pozwala na obserwacje żadnych silnych korelacji

pomiędzy tymi odpowiedziami. W przypadku największych zbiorów przetwarzanych danych (powyżej 50GB) występują zarówno analizy w skali lokalnej, pojedynczych gmin czy powiatów, jak i całego kraju.



Rysunek 28. Zestawienie odpowiedzi na pytania „Jak duży zakres przestrzenny danych przetwarzasz najczęściej?" (kolor pomarańczowy) oraz „Jak duży zakres przestrzenny danych przetwarzałbyś, jeżeli byłoby to możliwe?" (kolor zielony) w rozbiciu na zakresy analizy danych przestrzennych (opracowanie własne)

6.3. Wnioski z badań ankietowych

Wyniki ankiety prowadzą do wielu wniosków na temat narzędzi, problemów i wyzwań w przetwarzaniu danych przestrzennych (w szczególności SBD) w Polsce przez specjalistów GIS. Zdecydowana większość z nich sygnalizuje problemy z przetwarzaniem dużych zbiorów danych przestrzennych oraz oczekuje zwiększenia zakresu przestrzennego przetwarzanych danych. Jednocześnie, nadal najczęściej stosowanym narzędziem przez tych specjalistów są narzędzia klasy desktop GIS. Zauważalną popularność zdobyło też analizowanie danych przestrzennych poprzez bezpośrednie użycie języków programowania (w szczególności języka Python), często z użyciem interfejsu programistycznego Jupyter Notebook/Lab (lub podobnych). Najczęściej przetwarzanym typem danych są dane wektorowe. Bardzo często wykorzystywane jest jednak wiele typów danych na raz (w szczególności dane wektorowe razem z rastrowymi). Zauważyć

należy jednak dużą różnorodność w przetwarzanych typach danych i prowadzanych rodzajach analiz, a także szeroki zakres doboru narzędzi zarówno pod względem sprzętu jak i oprogramowania.

Wśród specjalistów GIS w Polsce nie są popularne narzędzia SBD, pomimo tego wielu z nich mierzy się z problemem przetwarzania dużych zbiorów danych i rozwiązuje go z użyciem technik takich jak podział danych na mniejsze części, optymalizacja używanych algorytmów czy zastosowanie przetwarzania równoległego. Potwierdza to postawioną na tym etapie badań hipotezę:

Eksperci analizy danych przestrzennych w Polsce napotykają i rozwiązują problemy analizy SBD bez bezpośredniego wykorzystania narzędzi SBD.

Niemal 40% respondentów korzysta z technologii chmurowych w swojej pracy, co wskazuje na wysoki poziom integracji tych technologii z GIS. Pomimo, iż ponad 60% odpowiadających przetwarza dane o wielkości do 25 GB co nie jest bardzo dużą wartością, to nawet w tej grupie większość deklaruje napotykanie problemów wynikających z wielkości zbioru. W grupie przetwarzającej dane o wolumenie 50GB i większym, wszyscy badani deklarują występowanie tego problemu.

Przeprowadzone badanie ankietowe obarczone jest ograniczeniami związanymi z niewielką liczbą uczestników oraz wąską specjalizacją respondentów. Większość ankietowanych związana jest z naukami inżynieryjno-technicznymi oraz ścisłymi i przyrodniczymi.

Przeprowadzone badania przyniosły jednak zdaniem autora przydatne w dalszej pracy odpowiedzi i pozwalają na ogólne scharakteryzowanie obecnej sytuacji dotyczącej sposobu podejścia do prowadzenia analiz geoprzestrzennych w Polsce.

Realnym i istotnym problemem jest obecnie zagadnienie przetwarzania dużych zbiorów danych przestrzennych, a wykorzystywane technologie potencjalnie ograniczają wydajność i zakres przestrzenny analiz geoprzestrzennych.

7. Luki badawcze

Przeprowadzona powyżej analiza literatury oraz badania ankietowe obrazują ogólny stan wykorzystania technologii SBD w praktyce przez specjalistów i naukowców z zakresu GIS. Po początkowej eksplozji zainteresowania w okolicach roku 2017, technologie te weszły w stan stabilnego rozwoju znajdując swoją niszę odbiorców. Powstał również szereg rozwiązań komercyjnych wspierających technologie SBD, czego przykładem może być Google Earth Engine. W Polsce wykorzystanie tych technologii przez specjalistów z branży jest niewielkie, mimo, iż ograniczania wydajnościowe stanowią dla nich istotne wyzwanie. Badania porównawcze zwykle nie uwzględniają klasycznych narzędzi GIS, przez co proces decyzyjny dotyczący wyboru narzędzi jest utrudniony. Wykorzystanie narzędzi SBD jest dodatkowo ograniczone koniecznością rozszerzenia wiedzy przez osoby, które chcą ich użyć. Technologie te jak dotąd nie są w standardzie faktycznym nauczania na kierunkach geodezyjno-kartograficznych, geograficznych, a nawet geoinformatycznych. Z drugiej strony problemy z przetwarzaniem bardzo dużych zbiorów danych są często omijane poprzez wykorzystanie dobrze znanego w branży oprogramowania, pomimo, iż może to wymagać dodatkowych nakładów pracy lub wykraczać poza możliwości stosowanych narzędzi. Same badania wydajności SBD najczęściej dotyczą stosunkowo niewielkich albo symulowanych zbiorów danych, co jest zupełnie wystarczające na potrzeby porównania wydajności rozwiązań, ale nie przynosi informacji o możliwych ograniczeniach tych metod w przypadku wykorzystania do przetworzeń na realnych, bardzo dużych zbiorach danych przestrzennych. Brakuje również zbiorów danych mogących stanowić *benchmark* dla różnych algorytmów oraz konkretnych cyberinfrastruktur danych przestrzennych.

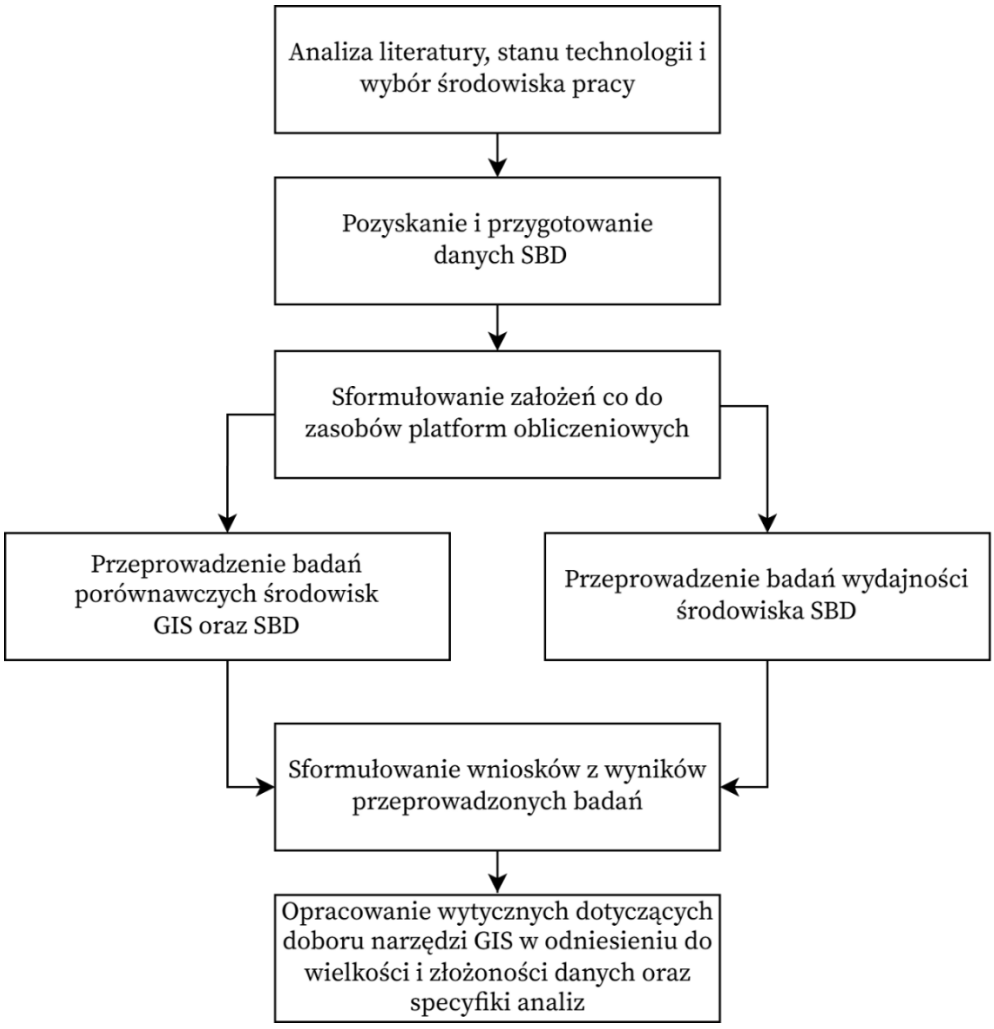
Podsumowując, autor rozprawy zidentyfikował następujące luki badawcze:

- Brak badań porównawczych zestawiających wydajność klasycznych narzędzi GIS z narzędziami SBD, w zależności od charakterystyki oraz wielkości zbioru danych
- Brak badań dotyczących wykorzystania narzędzi SBD do przeprowadzania analiz przestrzennych w skali całej Polski (z użyciem bardzo dużych zbiorów danych).

8. Metodyka badań

Mnogość dostępnych technologii przetwarzania danych przestrzennych (zarówno klasycznych, jak i SBD) oraz szeroki zakres analiz przestrzennych możliwych do wykonania nie pozwala na pełne odniesienie się do zidentyfikowanych powyżej luk badawczych w ramach jednej rozprawy doktorskiej. Konieczne było więc podjęcie decyzji ograniczających badania. Zdecydowano się sprowadzić dalsze badania do jednej cyberinfrastruktury danych przestrzennych (oraz powiązanego z nią jednego środowiska SBD) oraz przykładowych zbiorów danych i algorytmów.

W celu jak najdokładniejszego określenia wydajności metod SBD oraz klasycznych narzędzi GIS, badania oparto o metodę eksperymentalną. Na poniższym diagramie przedstawiono przyjęty schemat procesu badawczego.



Rysunek 29. Schemat ilustrujący ogólny przebieg (metodykę) badań (opracowanie własne).

Biorąc pod uwagę cel pracy, kluczowym aspektem jest wybór środowiska testowego. Analiza literatury oraz istniejących rozwiązań wykazała, że dostępnych jest niewiele środowisk obliczeniowych dla dużych danych przestrzennych, które umożliwiają prowadzenie zaawansowanych analiz z wykorzystaniem zróżnicowanych zbiorów danych przestrzennych. Nie istnieje również standard, w oparciu o który środowiska tego typu są budowane. W związku z czym nie jest aktualnie możliwa ocena wydajności technologii przetwarzania dużych danych przestrzennych jako całości. Pierwszym krokiem badań był więc dobór cyberinfrastruktury danych przestrzennych.

Po selekcji cyberinfrastruktury należało dobrać i pozyskać odpowiednie zbiory danych przestrzennych. Badania oparto o zbiór danych, który jest wystarczająco obszerny, aby jego przetworzenie w sposób klasyczny z użyciem pojedynczej maszyny obliczeniowej było bardzo utrudnione, a jednocześnie spełnia założenia „V" big data.

Właściwe badania przeprowadzono w rozbiciu na dwa etapy:

1. Badania porównawcze środowisk GIS oraz SBD
2. Badania wydajności środowiska SBD

Pierwszy etap badań miał za zadanie dostarczyć informacji dotyczących wydajności klasycznych narzędzi GIS względem metod SBD oraz określić pułap wielkości zbiorów, dla których zastosowanie tych narzędzi przestaje być optymalne z uwzględnieniem poziomu trudności obsługi narzędzi i wymaganych umiejętności programistycznych. Drugi etap badań skupiał się natomiast na określeniu wydajności środowiska SBD w analizie dużych zbiorów danych przestrzennych, a wnioski z tej części badań dostarczyły wiedzy z zakresu możliwości i ograniczeń środowiska w kontekście analiz przestrzennych w skali całego kraju. W przypadku obu etapów badań konieczne było określenie założeń co do wykorzystywanych zasobów obliczeniowych oraz metod pomiarów. Szczegółową metodykę obu etapów badań przedstawiono w dalszej części tego rozdziału.

Finalnie, wynikiem obu etapów badań były obserwacje oraz wnioski stanowiące podstawę do opracowania wytycznych dotyczących doboru narzędzi GIS/SBD w odniesieniu do wielkości i złożoności danych oraz specyfiki przeprowadzanych analiz przestrzennych. Ponieważ

badaniom podlegały wybrane zbiory danych i algorytmy, opracowane wytyczne mają charakter ogólnej heurystyki doboru narzędzi analiz przestrzennych.

8.1. Środowisko eksperymentalne CENAGIS

Wydajność i stabilność metod SBD jest silnie uzależniona od konkretnej infrastruktury w której zostały zaimplementowane. Wpływ na nią mają przede wszystkim czynniki takie jak: wersje oprogramowania, zastosowany sprzęt komputerowy (CPU, RAM, rodzaj dysków), infrastruktura sieciowa, zastosowany system plików czy wykorzystywane komponenty pomocnicze (m.in. orkiestrator kontenerów, proxy, system poświadczeń). Kluczowym zatem jest, aby w procesie badań wydajności metod SBD uwzględnić konkretną infrastrukturę informatyczną, w której zostały one przeprowadzone.

Przyjęto następujące założenia, które środowisko musi spełniać w celu poprawnego przeprowadzenia eksperymentów założonych w ramach badań:

1. Dostęp do technologii rozproszonego przetwarzania dużych danych przestrzennych różnych typów – wektorów, rastrów oraz chmur punktów
2. Dostęp do dużych zbiorów danych przestrzennych w skali całego kraju
3. Możliwość monitorowania aktualnego obciążenia środowiska w celu zapewnienia powtarzalności eksperymentów
4. Dostęp do dużych mocy obliczeniowych i możliwość sterowania aktualnie zajętymi zasobami
5. Możliwość wprowadzania zmian do środowiska oraz implementacji własnych algorytmów

W oparciu o przyjęte założenia bazowe oraz przegląd dostępnych cyberinfrastruktur danych przestrzennych, wyselekcjonowano środowisko obliczeniowe CENAGIS do przeprowadzenia badań. Spełnia ono wszystkie opisane powyżej założenia. W szczególności pozwala na dużą elastyczność w zakresie wykorzystywanych mocy obliczeniowych oraz wprowadzania zmian w środowisku co nie byłoby możliwe w przypadku dedykowanych środowisk komercyjnych. Bardzo ważnym argumentem był też bezpłatny i nieograniczony dla niniejszej rozprawy

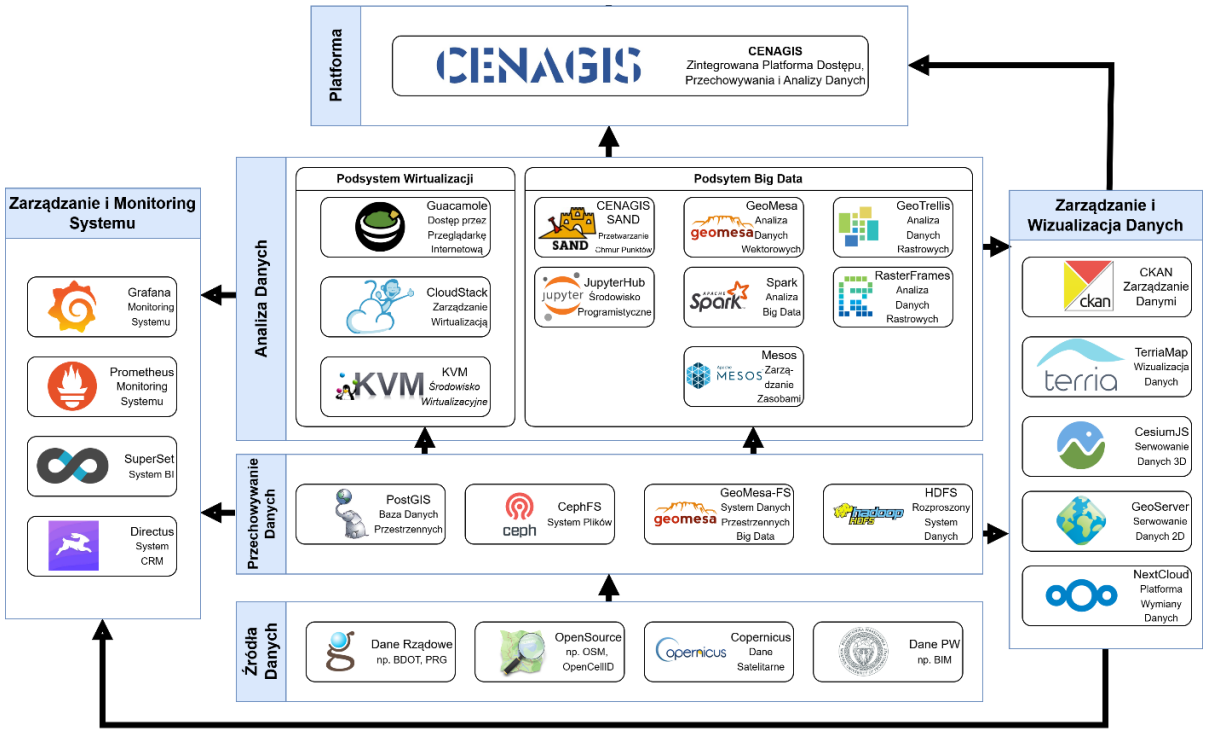
doktorskiej dostęp do Platformy IT CENAGIS oraz konkretne praktyczne potrzeby jej przetestowania. Stanowi ona obecnie jedną z najbardziej kompleksowych infrastruktur dostępnych dla celów badawczych i jest szeroko wykorzystywana przez naukowców z całej Polski. Dlatego jej wybór jako środowisko testowe był naturalny i uzasadniony merytorycznie.

Należy jednak pamiętać, że część wniosków wynikłych z procesów optymalizacji i analiz wydajności zamieszczonych w dalszych rozdziałach dotyczyć będzie właśnie tego konkretnego środowiska i nie będzie możliwe pełne przeniesienie wniosków na inne infrastruktury różniące się parametrami sprzętowymi, wersjami bibliotek czy innym podejściem do przydzielania mocy obliczeniowych.

CENAGIS to akronim od nazwy CEntrum Naukowych Analiz Geoprzestrzennych i Satelitarnych. Jego kluczowym elementem jest cyberinfrastruktura danych przestrzennych zbudowana na Wydziale Geodezji i Kartografii Politechniki Warszawskiej we współpracy z partnerami przemysłowymi, w szczególności z firmą OPEGIEKA[37]. Została zaprojektowana w celu prowadzenia badań nad przetwarzaniem bardzo dużych zbiorów danych przestrzennych oraz wspomagania prac R&D w geoinformacji. Platforma bazuje na infrastrukturze informatycznej umożliwiającej zdalny dostęp do dużych mocy obliczeniowych oraz zasobów danych przestrzennych. Na cyberinfrastrukturę CENAGIS (zwaną formalnie Platformą IT CENAGIS) składają się dwa klastry obliczeniowe nazwane na część wybitnych kartografów – MERCATOR oraz ROMER. Klaster MERCATOR składa się z dwóch Podsystemów – Wirtualizacji oraz Big Data. Pierwszy z nich dysponuje mocą całkowitą na poziomie 1092 vCPU, 4.2 TB RAM, 365 TB przestrzeni na dysku twardym oraz trzech jednostek GPU: dwóch NVIDIA V100 oraz jednej NVIDIA A100. Podsystem Big Data dysponuje mocą 1808 vCPU, 10.6 TB RAM, 2.5 PB przestrzeni na dysku w ramach rozproszonego systemu plików HDFS oraz sześćdziesięciu jednostek GPU NVIDIA T4. Klaster ROMER stanowi węzeł rozwojowy infrastruktury i dysponuje zasobami na poziomie 624 vCPU, 2.4 TB RAM i 346 TB przestrzeni na dysku twardym (Gotlib, Choromański i Kaczałek, 2022). Na infrastrukturę CENAGIS składają się także fizyczne laboratoria testowania urządzeń i aplikacji, znajdujące się w Józefosławiu. Niniejszy rozdział będzie skoncentrowany jednak na opisie infrastruktury informatycznej CENAGIS.

[37] https://opegieka.pl (data dostępu 15.12.2025)

System IT CENAGIS składa się z wielu komponentów połączonych w celu zapewnienia możliwie szerokich możliwości analizy przestrzennej na potrzeby badań. Narzędzia wykorzystane w poszczególnych elementach systemu przedstawiono na rysunku 30.



Rysunek 30. Narzędzia, dane i platformy informatycznej CENAGIS w podziale na jej poszczególne komponenty (opracowanie własne)

Architektura i funkcjonalność CENAGIS zostały zaprojektowane tak, aby wspierać wszystkie etapy pracy z danymi geoprzestrzennymi - od pozyskiwania i przechowywania, poprzez przetwarzanie i analizę, aż po udostępnianie wyników. System został zbudowany z wykorzystaniem otwartych i modularnych komponentów, które współdziałają w ramach jednolitego środowiska, umożliwiając użytkownikom wykonywanie zarówno klasycznych analiz GIS, jak i zaawansowanego przetwarzania danych w modelu big data.

U podstaw platformy znajdują się mechanizmy odpowiedzialne za pozyskiwanie i gromadzenie danych przestrzennych pochodzących z licznych źródeł, takich jak dane instytucji publicznych, zasoby otwarte, dane satelitarne oraz zbiory generowane wewnętrznie przez użytkowników platformy. Zgromadzone dane są przechowywane w środowiskach magazynowania pod różnymi postaciami w zależności od ich specyfiki. Dostępne rozwiązania obejmują relacyjne

bazy danych przestrzennych (np. PostGIS[38]), systemy plików (np. HDFS[39], CephFS[40]), a także specjalizowane rozwiązania dla danych typu SBD (Accumulo[41], GeoMesa-FS[42]).

Metody SBD zostały zaimplementowane na Platformie IT CENAGIS w ramach Podsystemu Big Data. Oparty jest on o środowisko Apache Spark, pozwalające na rozproszone przetwarzania danych na wielu maszynach na raz. To narzędzie rozszerzono następnie o pakiety ułatwiające analizę danych przestrzennych. Do przechowywania i analizy danych wektorowych wykorzystano oprogramowanie GeoMesa. Na potrzeby danych rastrowych zintegrowano środowisko GeoTrellis i RasterFrames. Wydajne przetwarzanie chmur punktów wspiera autorska biblioteka CENAGIS.SAND. Jest ona jednak tylko częściowo zintegrowana ze środowiskiem Spark. Dostęp do środowiska oparty jest o technologię JupyterHub[43], a Apache Mesos[44] odpowiada za zarządzanie konteneryzacją, na której oparte jest działanie tej części systemu.

Wykorzystanie klasycznych środowisk GIS w środowisku CENAGIS możliwe jest z użyciem podsystemu wirtualizacji. W tym celu wykorzystano silnik wirtualizacji KVM, a do zarządzania zasobami użyto oprogramowania CloudStack[45]. Dostęp do maszyn wirtualnych z poziomu przeglądarki internetowej umożliwia rozwiązanie Guacamole[46].

Uzupełnieniem całego środowiska są narzędzia do zarządzania i wizualizacji danych. Obejmują one zarówno systemy katalogowania zbiorów danych (CKAN[47]), jak i serwery mapowe i platformy do wizualizacji wyników analiz w środowiskach 2D i 3D (GeoServer[48], CesiumJS[49], TerriaMap[50]). Komponent NextCloud[51] umożliwia współdzielenie danych. Monitoring i kontrola

[38] https://postgis.net (data dostępu 15.12.2025)
[39] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (data dostępu 15.12.2025)
[40] https://docs.ceph.com/en/latest/cephfs (data dostępu 15.12.2025)
[41] https://accumulo.apache.org (data dostępu 15.12.2025)
[42] https://www.geomesa.org/documentation/stable/user/filesystem/index.html (data dostępu 15.12.2025)
[43] https://jupyter.org/hub (data dostępu 15.12.2025)
[44] https://mesos.apache.org (data dostępu 15.12.2025)
[45] https://cloudstack.apache.org (data dostępu 15.12.2025)
[46] https://guacamole.apache.org (data dostępu 15.12.2025)
[47] https://ckan.org (data dostępu 15.12.2025)
[48] https://geoserver.org (data dostępu 15.12.2025)
[49] https://cesium.com/platform/cesiumjs (data dostępu 15.12.2025)
[50] https://app.terria.io/terria/catalog/map (data dostępu 15.12.2025)
[51] https://nextcloud.com (data dostępu 15.12.2025)

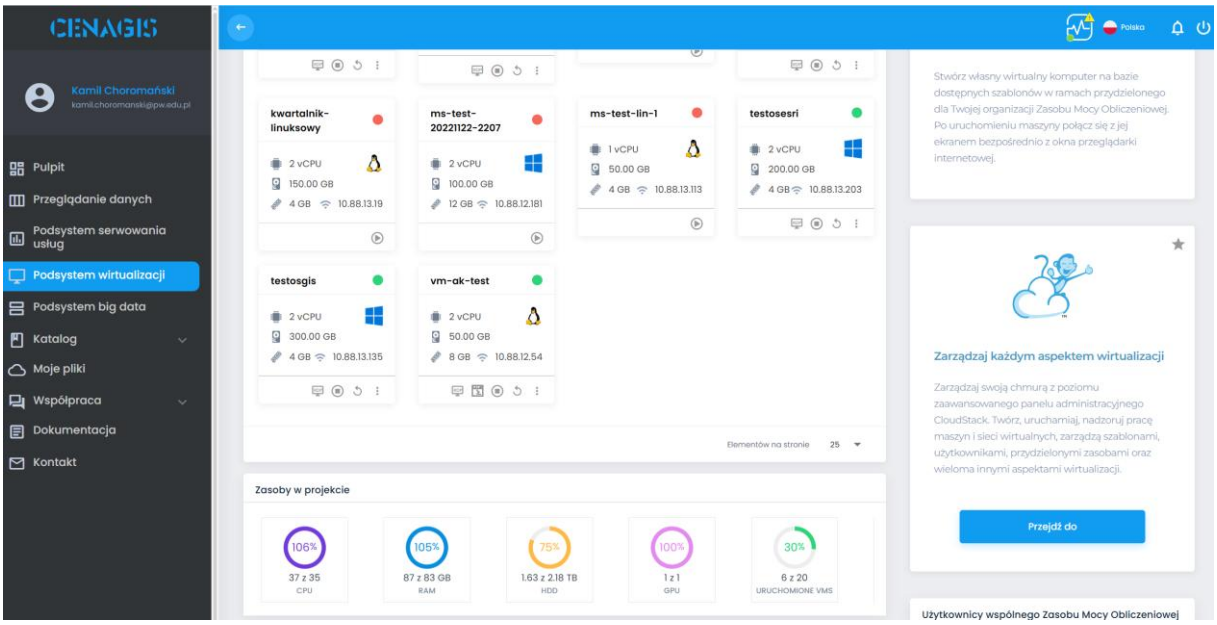
wydajności platformy realizowane są za pomocą narzędzi takich jak Prometheus[52], Grafana[53] czy Superset[54], które zapewniają przejrzystość działania klastra i ułatwiają zarządzanie operacyjne.

Dostęp do poszczególnych podsystemów i funkcji platformy zintegrowany jest przez panel CENAGIS oraz system logowania SSO (ang. Single Sign-On). W poniższych podrozdziałach przedstawiono szczegóły dotyczące działania obu podsystemów wykorzystanych do przeprowadzania badań – Podsystemu Wirtualizacji oraz Podsystemu Big Data.

8.1.1. Podsystem Wirtualizacji

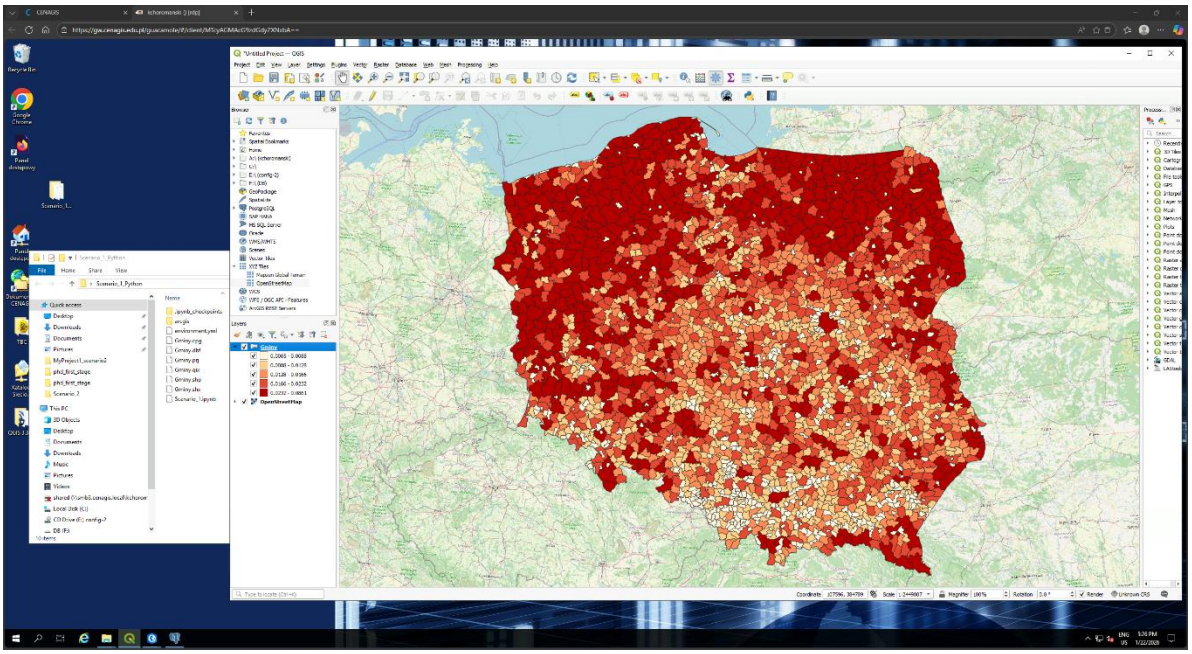
Na podsystem wirtualizacji składają się serwery fizyczne z systemem operacyjnym CentOS, a za proces wirtualizacji zasobów tych maszyn odpowiada silnik KVM. Zarządzanie środowiskiem wirtualizacji obsługiwane jest przez oprogramowanie CloudStack. Dla użytkowników Podsystemu dostępny jest panel CloudStack pozwalający na tworzenie własnych maszyn wirtualnych o zadanym systemie operacyjnym i mocach obliczeniowych. Moce te podzielone są na projekty, a do każdego projektu może być przydzielony jeden lub więcej użytkowników. Każdy z użytkowników może dowolnie rozporządzać mocami obliczeniowymi w ramach każdego z projektów do którego należy. Środowisko CloudStack zarządza tworzeniem maszyn wirtualnych i konfiguracją ustawień sieciowych na konkretnych serwerach fizycznych, dzięki czemu z punktu widzenia użytkownika tworzenie maszyny wirtualnej jest procesem sprawdzającym się do wybrania nazwy maszyny, systemu operacyjnego oraz mocy obliczeniowych. Dodatkowo, w ramach CENAGIS wytworzono dedykowany panel zarządzania wirtualizacją, który jeszcze mocniej upraszcza ten proces (rysunek 31).

[52] https://prometheus.io (data dostępu 15.12.2025)
[53] https://grafana.com (data dostępu 15.12.2025)
[54] https://superset.apache.org (data dostępu 15.12.2025)



Rysunek 31. Panel podsystemu wirtualizacji CENAGIS dostępny do zalogowaniu do systemu na stronie panel.cenagis.edu.pl

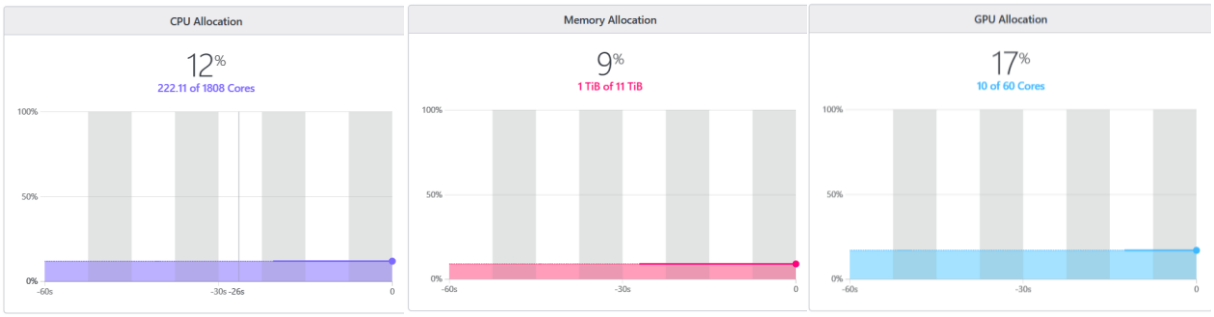
Istnieje możliwość utworzenia maszyny wirtualnej zarówno z systemem operacyjnym Linux (Ubuntu, Rocky Linux) jak i Windows Server. Przygotowane są również szablony specjalizowane pod konkretne zastosowania (np. przetwarzanie fotogrametryczne, analiza GIS z użyciem narzędzi *open source* czy analiza GIS z użyciem narzędzi komercyjnych jak ESRI ArcGIS czy Hexagon GeoMedia). W dalszym kroku użytkownik może zdecydować o liczbie vCPU, ilości pamięci RAM, lokalnych zasobach dyskowych czy dostępie do jednostek GPU. Po utworzeniu maszyny możliwe jest połączenie się z nią z wykorzystaniem wirtualnego pulpitu lub połączenia SSH (w przypadku maszyn z systemem operacyjnym Linux). W tym celu wykorzystywane jest oprogramowanie Apache Guacamole, które umożliwia ten dostęp z poziomu przeglądarki internetowej bez potrzeby instalacji dodatkowego oprogramowania typu VPN. Praca z samą maszyną wirtualną przypomina pracę ze standardowym komputerem osobistym klasy desktop. Poniżej przedstawiono przykład widoku z poziomu zdalnego pulpitu dla maszyny z systemem Windows z uruchomionym oprogramowaniem QGIS.



Rysunek 32. Przykładowy widok pulpitu zdalnego maszyny wirtualnej z systemem Windows z uruchomionym oprogramowaniem QGIS (opracowanie własne)

8.1.2. Podsystem Big Data

Podsystem Big Data opiera swoje działanie na środowisku Apache Mesos, które umożliwia zarządzanie środowiskiem konteneryzacji z użyciem wielu serwerów na raz. Do obsługi klastra zintegrowano oprogramowanie DC/OS[55] (ang. Distributed Cloud Operating System), które pozwala na zarządzanie i wizualizację zasobów dostępnych w ramach Podsystemu Big Data. Na rysunku 33 przedstawiono fragment panelu DC/OS pozwalający na obserwację aktualnego zużycia zasobów w podsystemie.



Rysunek 33. Zestawienie zasobów podsystemu big data dostępne w interfejsie DC/OS (opracowanie własne)

Dynamiczne skalowanie mocy obliczeniowych jest zapewnione przez rozwiązanie Marathon, stanowiące część ekosystemu Mesos. Możliwość dynamicznego przydzielania mocy

[55] https://dcos.io (data dostępu: 15.12.2025)

obliczeniowych jest konieczne dla obliczeń wykonywanych z użyciem Apache Spark, który to stanowi silnik obliczeń rozproszonych. Podsystem Big Data wykorzystuje wersje 2.4.7 Spark zmodyfikowaną w taki sposób, aby możliwa była pełna integracja z bibliotekami analiz przestrzennych tj. GeoMesa 3.1.1, RasterFrames 0.9.1 oraz geopyspark (nakłada języka Python na GeoTrellis) 0.4.5. Głównym językiem programowania wykorzystywanym w Podsystemie Big Data jest język Python 3.7.

Na komponenty klastra Spark składają się program sterujący (tzw. *driver*, ang. *driver program*), zarządca klastra (ang. *cluster manager*) oraz jednostki wykonujące polecenia (tzw. *worker*, ang. *worker node*). Program sterujący jest aktywowany poprzez zainicjalizowanie kontekstu Spark z zadeklarowaną maksymalną liczbą wirtualnych procesorów oraz ilością pamięci RAM, które mają być przydzielone w ramach maszyn *worker*. Moce te zostaną zarezerwowane dopiero przy rozpoczęciu obliczeń, a ich zwolnienie do puli nastąpi po ich zakończeniu. Polecenia dla klastra Spark mogą być definiowane z użyciem środowiska programistycznego pySpark oraz w języku zapytań SparkSQL. Są one następnie interpretowane do postaci zadań delegowanych do poszczególnych maszyn *worker*.

Proces interpretacji poleceń na zadania wykonywany jest z użyciem modułu Catalyst Optimizer, którego zadaniem jest optymalizacja zapytań z uwzględnieniem całości zleconych obliczeń (Michael i inni, 2015). Optymalizacja wielu poleceń obliczeniowych składających się na całość obliczeń jest możliwa dzięki zastosowaniu mechanizmu wartościowania leniwego (ang. *lazy evaluation*). Podejście to zakłada realizacje zleconych operacji dopiero w momencie, gdy jest to konieczne (np. w momencie zapisu danych na dysku lub wyświetlenia wyniku), co pozwala na optymalizację całej serii operacji. Takie polecenie jest interpretowane do postaci planu logicznego wykonania operacji, który jest następnie optymalizowany i reprezentowany w postaci grupy planów fizycznych (serii niskopoziomowych instrukcji dla silnika Spark). Optymalny plan fizyczny jest następnie selekcjonowany i przekazywany do wykonania (Michael i inni, 2015). Spark pozwala na wyświetlanie planów logicznych i fizycznych, a ich interpretacja ułatwia zrozumienie powodów problemów wydajnościowych zleconych obliczeń.

Kolejnym istotnym aspektem środowiska Spark jest sposób operowania na danych. Fundamentalną abstrakcję przetwarzania danych w Apache Spark stanowi RDD (Resilient Distributed Dataset), czyli rozproszony, odporny na błędy zbiór danych podzielony na partycje

i przetwarzany równolegle na wielu węzłach obliczeniowych. RDD stanowi bazowy model obliczeniowy całego środowiska Spark. W praktyce jednak, w większości współczesnych zastosowań, podstawową abstrakcją wykorzystywaną do pracy z danymi strukturalnymi jest DataFrame (ramka danych), który można traktować jako rozwinięcie koncepcji RDD o jawnie zdefiniowany schemat danych oraz mechanizmy optymalizacji zapytań. DataFrame reprezentuje dane w postaci tabelarycznej, zbliżonej do relacyjnego modelu danych, co upraszcza operacje analityczne i umożliwia efektywniejsze planowanie wykonania przez silnik Spark SQL.

Wewnętrznie DataFrame, podobnie jak RDD, dzielony jest na partycje, które stanowią wydzielone podzbiory danych przetwarzane w ramach poszczególnych maszyn *worker*. Spark umożliwia pełną kontrolę nad liczbą partycji w ramach jednego DataFrame oraz metodą ich podziału. Partycje mogą być organizowane w taki sposób, aby zawierały wspólne wartości lub określone zakresy wartości w wybranych kolumnach, co ma istotne znaczenie z punktu widzenia lokalności danych oraz równoważenia obciążenia.

Pomimo, iż środowisko Spark posiada wbudowanych wiele algorytmów (w tym z zakresu analizy grafowej czy uczenia maszynowego), to dostępność gotowych i sprawdzonych algorytmów jest znacznie mniejsza niż np. w przypadku bibliotek języka Python. Możliwe jest jednak uniknięcie czasochłonnej i narażonej na błędy ręcznej implementacji z wykorzystaniem mechanizmu pandas User-Defined Functions (pandas UDF). Umożliwia on wykonanie po stronie maszyn *worker* kodu języka Python. W wypadku tych funkcji poszczególne partycje Spark DataFrame są przetwarzane do postaci obiektu DataFrame biblioteki pandas.

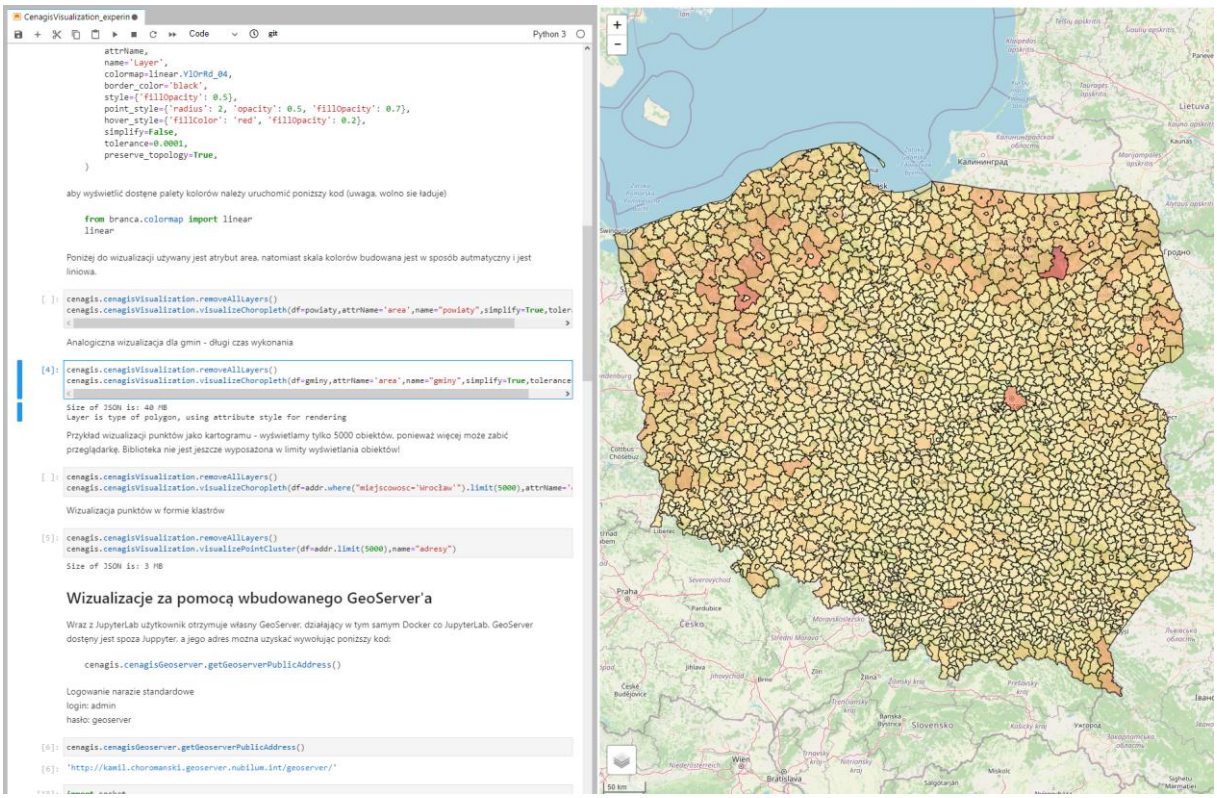
Biblioteki GeoMesa, RasterFrames oraz GeoTrellis zapewniają dostęp do funkcji analizy przestrzennej w ramach Spark. Biblioteka GeoMesa odpowiada za analizę danych wektorowych. Udostępnia ona algorytmy pozwalające na analizę relacji przestrzennych, transformacje układu współrzędnych, złączenia przestrzenne czy modyfikacje geometrii. Biblioteki RasterFrames oraz GeoTrellis umożliwiają analizę danych rastrowych poprzez podział rastra na kafle, które są reprezentowane w ramach pojedynczego obiektu Spark DataFrame, co pozwala na prowadzenie obliczeń rozproszonych. Biblioteki te posiadają funkcje algebry rastrów, analizy sąsiedztwa czy złączeń rastrów.

Przechowywanie danych w sposób rozproszony oparte zostało o system plików Hadoop (ang. Hadoop Distribiuted File System – HDFS) oparty o wersję 3.1.4 Apache Hadoop. Głównym założeniem HDFS jest przechowywanie dużych zbiorów danych w sposób bezpieczny oraz wydajny z nastawieniem na operacje rozproszone. Podstawową jednostką przechowywania danych w HDFS są bloki danych – pakiety o określonej wielkości. W postaci bloków danych przechowywane są wszystkie pliki ładowane na HDFS. Bloki są następnie replikowane na różne maszyny klastra HDFS w celu zapewnienia bezpieczeństwa danych w razie awarii dysków. Z punktu widzenia użytkownika HDFS posiada interfejs zbliżony do standardowego systemu plików systemach Linux.

Przechowywanie danych przestrzennych w rozproszonym systemie plików możliwe jest na kilka sposobów. W ramach pakietu GeoMesa udostępniono także możliwość zapisu danych przestrzennych z zastosowaniem indeksowania. Pozwala to na optymalizacje obliczeń poprzez selektywny odczyt tylko potrzebnych do obliczeń danych z dysków. Indeksowanie odbywa się z wykorzystaniem algorytmu krzywej wypełniającej przestrzeń Mortona. Transformuje ona dane z postaci wielowymiarowej do jednego wymiaru (ciągu liczb) pozwalającego na jednoznaczną identyfikacje danego obszaru (Morton, 1966). GeoMesa pozwala na zdefiniowanie rozdzielczości przestrzennej indeksu, a także na indeksowanie danych przestrzennych z dodatkowym komponentem czasowym. Taki zbiór danych może być zapisany w postaci struktury GeoMesa-FS lub w bazie danych Accumulo. Plik GeoMesa-FS zapisywany jest bezpośrednio na HDFS i wewnętrznie składa się z plików Parquet[56] o odpowiednich nazwach z zakodowanym indeksowaniem przestrzennym lub czasowo-przestrzennym. W drugim przypadku rekord zapisywany jest w strukturze bazy danych typu klucz-wartość, gdzie indeks przestrzenny zapisywany jest jako część klucza. Dane przechowywane w ramach bazy Accumulo są również przechowywane na HDFS. Format Parquet stanowi popularny format przechowywania dużych zbiorów danych ze względu na swoją kolumnową architekturę. Umożliwia to znaczną kompresję danych (szczególnie dla danych o wysokiej redundancji) przy zachowaniu krótkiego czasu odczytu. Format ten cechuje się wysoką wydajnością w realnych zadaniach obliczeniowych (Zeng i inni, 2023).

[56] Kolumnowy format danych rozwijany na zasadach *open source* (https://Parquet.apache.org, data dostępu 15.12.2025)

Elementem systemu odpowiedzialnym za dostęp do środowiska jest JupyterHub. Rozwiązanie to pozwala na dynamiczne tworzenie środowisk programistycznych Jupyter w środowiskach chmurowych. W CENAGIS interfejs Jupyter używany jest jako punkt dostępowy do Podsystemu Big Data i z jego poziomu możliwe jest inicjalizowanie obliczeń SBD z użyciem klastra. Sam interfejs dostępny jest z poziomu przeglądarki internetowej oraz pozwala na łączenie kodu, elementów opisowych i graficznych w ramach jednego pliku (notatnika Jupyter). Środowisko Jupyter znajduje zastosowanie zarówno do prac badawczych i rozwojowych, jak i edukacji, a poprzez możliwość integracji z interaktywnymi mapami i szerokimi możliwościami wizualizacji danych przestrzennych może stanowić również narzędzie do analizy GIS (Choromański, Gotlib i Łobodecki, 2023). Poniżej przedstawiono przykładowy widok środowiska Jupyter do pracy z danymi przestrzennymi.



Rysunek 34. Środowisko Jupyter w Podsystemie Big Data CENAGIS skonfigurowane do pracy z danymi przestrzennymi (opracowanie własne)

8.2. Zbiory danych

W tym rozdziale przedstawiono charakterystykę zbiorów danych wykorzystanych do przeprowadzenia badań. Przeprowadzone analizy opierają się zarówno na danych *spatial big*

data, jak i na szeregu dodatkowych zbiorów danych przestrzennych pełniących rolę danych pomocniczych. Rozdział ma charakter przeglądowy i koncentruje się na ogólnym opisie wykorzystywanych danych, ich ogólnej strukturze, zakresie przestrzennym i czasowym oraz roli w całym procesie badawczym. Szczegółowe informacje dotyczące przygotowania danych, ich przetwarzania oraz sposobu wykorzystania w poszczególnych eksperymentach badawczych zostały przedstawione w rozdziałach opisujących scenariusze badawcze w rozdziałach 9 i 10.

8.2.1. Zbiór danych o cechach 5V

Do przeprowadzenia badań wydajnościowych SBD konieczne było pozyskanie zbioru danych nie tylko charakteryzującego się cechami „V" *big data*. Przyjęto, że zbiór musi być na tyle duży, aby możliwe było przetestowanie metod SBD w skali przewyższającej możliwości obliczeniowe pojedynczego komputera. Na potrzeby spełniania powyższych założeń wykorzystano dostępny w Politechnice Warszawskiej zbiór danych o mobilności pojazdów zakupiony od firmy Neptis (obecnie Yanosik SA), będącej producentem bardzo popularnej na polskim rynku aplikacji nawigacyjno-lokalizacyjnej Yanosik[57] z funkcją społecznościowych powiadomień o różnych zdarzeniach drogowych. Jest to zbiór danych typu FCD (ang. *floating car data*) z aplikacji Yanosik dla dwóch lat: 2019 i 2020.

Tego typu dane o mobilności oraz transporcie są często wykorzystywane w analizach SBD. Wskazują na to prace przeglądowe skupiające się na zagadnieniach dotyczących dużych danych o mobilności (Sakr, Ray i Renso, 2022) oraz wykorzystaniu analiz SBD w systemach transportowych (Zhang i Song, 2024). Istnieją też przykłady prac, gdzie analizowane są dane FCD z użyciem różnorodnych metod obliczeniowych (Ruzicka, Tichy i Hajciarova, 2022).

W tym zakresie, istnieją badania wykorzystujące narzędzia SBD (Apache Spark z rozszerzeniem Apache Sedona) do analizy prędkości pojazdów i przypisywania ich lokalizacji do odcinków dróg w oparciu o dane FCD (tzw. *map-matching*) (Zhang, Stewart i Li, 2023). Badanie to wykorzystuje stosunkowo duży zbiór danych (126 milionów punktów pomiarowych), których przypisanie do odpowiadających odcinków sieci drogowej (3.4 miliona segmentów) w stanie Kalifornia zajęło 7h 45 minut przy zastosowaniu 100 vCPU.

[57] https://yanosik.pl (data dostępu: 15.12.2025)

Jest to jeden z nielicznych przykładów kompleksowego zastosowania narzędzi SBD do przetwarzania dużych zbiorów danych o mobilności opisywanych w pracach badawczych. Pozyskany zbiór stanowi zatem reprezentatywny przykład danych klasy SBD oraz wpisuje się w aktualne kierunki rozwoju badań w tym obszarze.

Omawiany zbiór, pozyskany od firmy Neptis, zawiera dane w dwóch postaciach:

- Lokalizacje punktowe pojazdów – reprezentujące punktowe pozycje pojazdów pozyskane przez odbiorniki GNSS smartfonów z włączoną aplikacją połączone w trasy z informacją o ich aktualnej prędkości i kierunku
- Prędkości na odcinkach dróg – średnia prędkość przypisana do segmentów dróg (sieć OpenStreetMaps) w interwale 5 minut

Przykład rekordów ze zbioru lokalizacji punktowych pojazdów został przedstawiony na rysunku 35.

track_id	longitude	latitude	time	course	speed	segment_id	speed_limit
19110000003059	21.9866	50.04577	2019-01-01 00:00:01	105.0	25.0	-1	-1
19110000003059	21.98629	50.04545	2019-01-01 00:00:11	102.0	21.0	-1	-1
19110000003059	21.98596	50.04502	2019-01-01 00:00:21	62.0	7.0	-1	-1
19110000003059	21.98587	50.04506	2019-01-01 00:00:55	110.0	3.0	7726191	50
19110000001311	16.85346	52.36247	2019-01-01 00:00:00	19.0	100.0	2361552	100
19110000001311	16.85647	52.36434	2019-01-01 00:00:10	24.0	104.0	753155	100
19110000001311	16.85928	52.36643	2019-01-01 00:00:20	16.0	115.0	6746838	100
19110000001311	16.86191	52.36877	2019-01-01 00:00:30	16.0	111.0	6746838	100

Rysunek 35. Przykładowe rekordy ze zbioru danych lokalizacji punktowych pojazdów pozyskanych z aplikacji Yanosik (opracowanie własne)

Zbiór ten zawiera następujące kolumny:

- *track_id* – unikalny identyfikator pojedynczej podróży
- *longitude* – długość geograficzna
- *latititude* – szerokość geograficzna
- *time* – czas pomiaru
- *course* – azymut, w którym porusza się samochód
- *speed* – prędkość pojazdu w danym czasie (w km/h)

- *segment_id* – identyfikator segmentu drogi OpenStreetMaps (w przypadku braku zidentyfikowanego poprawnie segmentu występuje wartość „-1”)
- *speed_limit* – ograniczenie prędkości obowiązujące na powiązanym segmencie drogi (w km/h)

Na rysunku 36 zaprezentowano przykładowe rekordy ze zbioru danych o prędkościach na odcinkach dróg:

segment_id	dtg	avg_speed	method
87301	2019-01-02 19:40:00	28	5
87301	2019-01-02 19:45:00	50	2
87301	2019-01-02 19:50:00	30	1
87301	2019-01-02 19:55:00	35	5
87301	2019-01-02 20:00:00	35	5
87301	2019-01-02 20:05:00	35	5
87301	2019-01-02 20:10:00	35	5

Rysunek 36. Przykładowe rekordy ze zbioru danych prędkości na odcinkach dróg pozyskanych z aplikacji Yanosik (opracowanie własne)

Zbiór zawiera następujące kolumny:

- *segment_id* – identyfikator segmentu OSM
- *dtg* – czas do którego zagregowano informacje o prędkościach
- *avg_speed* – średnia prędkość na odcinku w zadanym czasie
- *method* – identyfikator metody obliczenia średniej prędkości

Identyfikator metody (a co za tym idzie sposób wyznaczania prędkości średniej) jest określany według poniższych warunków:

- 1: z okna 5. minutowego,
- 2: z okna 10. Minutowego
- 3: z okna 15. Minutowego
- 4: z okna 20. Minutowego

- 5: na podstawie innych dni roboczych w danej godzinie (dla dnia roboczego) - okres nie dłuższy niż miesiąc
- 6: na podstawie innych dni weekendowych w danej godzinie (dla dni weekendowych) - okres nie dłuższy niż miesiąc,
- 7: dla tej samej godziny, ale z dowolnego dnia (roboczego/weekendowego) i dłuższego okresu niż miesiąc.

Prędkość średnia wyliczana jest metodą o wyższym identyfikatorze tylko jeżeli nie zarejestrowano pomiarów przy użyciu metody o niższym identyfikatorze. W tabeli 9 poniżej przedstawiono zestawienie wielkości tych zbiorów danych w oryginalnej postaci plików CSV:

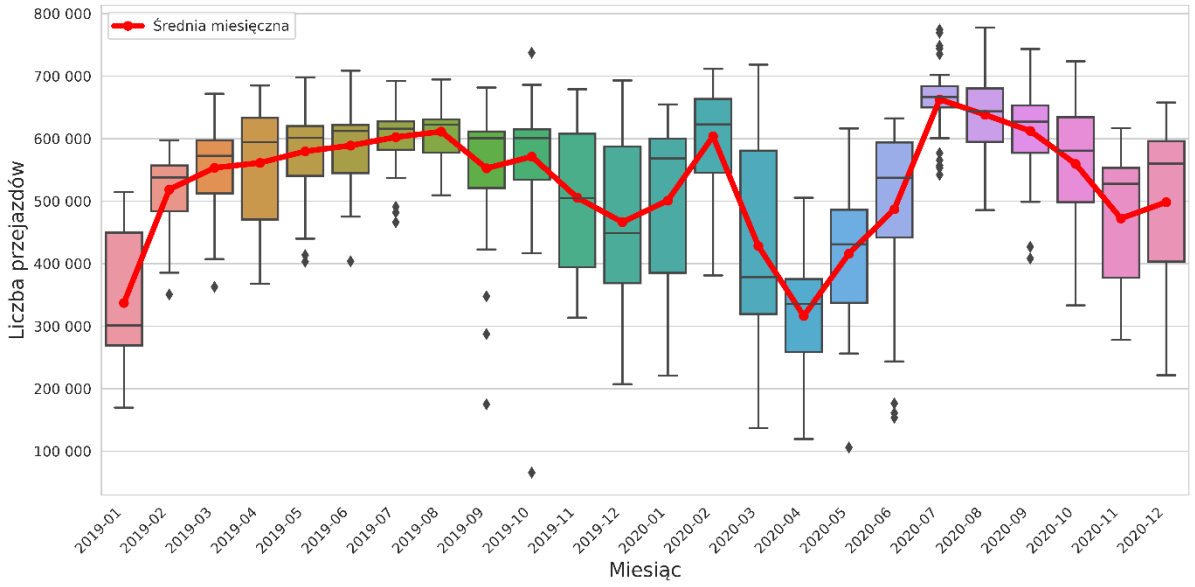
Tabela 9. Podsumowanie wielkości zbiorów danych pozyskanych z aplikacji Yanosik (opracowanie własne)

Zbiór	Wielkość (pliki CSV)
Średnie prędkości na odcinkach	14.8 TB
Lokalizacja i prędkość samochodów	6.7 TB
Łącznie	**21.5 TB**

Zbiór danych w oryginalnej postaci zajmuje łącznie 21.5 TB. Zbiór o średnich prędkościach na odcinkach zawiera 423 510 301 202 rekordów (423,5 miliarda), a zbiór o lokalizacji pojazdów 102 653 649 818 rekordów (102,6 miliarda). Są to wielkości znacznie przewyższające te wykorzystane w dotychczasowych badaniach porównawczych SBD czy te użyte w badaniach wykorzystujących dane FCD

Dane mają również dodatkową wartość informacyjną ze względu na czas ich akwizycji. Początek roku 2020 przypada na rozpoczęcie się pandemii SARS-CoV-2 w Polsce, przez co charakterystyka mobilności znacznie się zmieniła w związku ze stosowaniem różnego rodzaju obostrzeń w przemieszczaniu się. Jest to również bardzo dobrze widoczne w danych i pozwala analizować zmiany w mobilności związane z wprowadzeniem obostrzeń. Na rysunku 37 przedstawiono średnią dzienną liczbę pojazdów w podziale na miesiące. W okresie od marca do czerwca 2020 w analizowanych danych wyraźnie widać spadek średniej liczby dziennych przejazdów, co koresponduje z rządowymi restrykcjami przemieszczania się wprowadzonymi w czasie pierwszej fali pandemii. Zjawisko to potwierdzają także inne badania m.in.

(Wielechowski, Czech i Grzęda, 2020), a wykorzystanie w podobnych analizach tak rozległego zbioru danych mogłoby potencjalnie przynieść nowe wnioski również na tym polu.



Rysunek 37. Wykres uśrednionej średniej dziennej liczby przejazdów w podziale na miesiące na podstawie danych z aplikacji Yanosik (opracowanie własne)

Analizowane dane spełniają więc zarówno wszystkie cechy 5V charakterystyki *big data*:

1) Wielkość zbioru jest bardzo duża (*volume*)

2) Dane tego typu powstają w sposób ciągły (*velocity*)

3) Dane posiadają liczna atrybuty oraz szeroki kontekst czasowy i przestrzenny (*variety*)

4) Dane posiadają wysoką wartość informacyjną na temat rzeczywistych wzorców mobilności oraz ich zmian spowodowanych pandemią (*value*)

5) Dane pochodzą z odbiorników GNSS smartfonów, cechują się więc zróżnicowaną dokładnością (*veracity*)

Wolumen danych znacząco przewyższa zdecydowaną większość zbiorów danych wykorzystywanych do badań technologii SBD opisanych w rozdziale 2.3 (tabela 3). Powoduje to, że są one adekwatnym zbiorem do przeprowadzenia badań nad wydajnością i optymalizacją metod SBD.

8.2.2. Klasyczne zbiory danych

Inne, wykorzystanie w pracy zbiory danych obejmują:

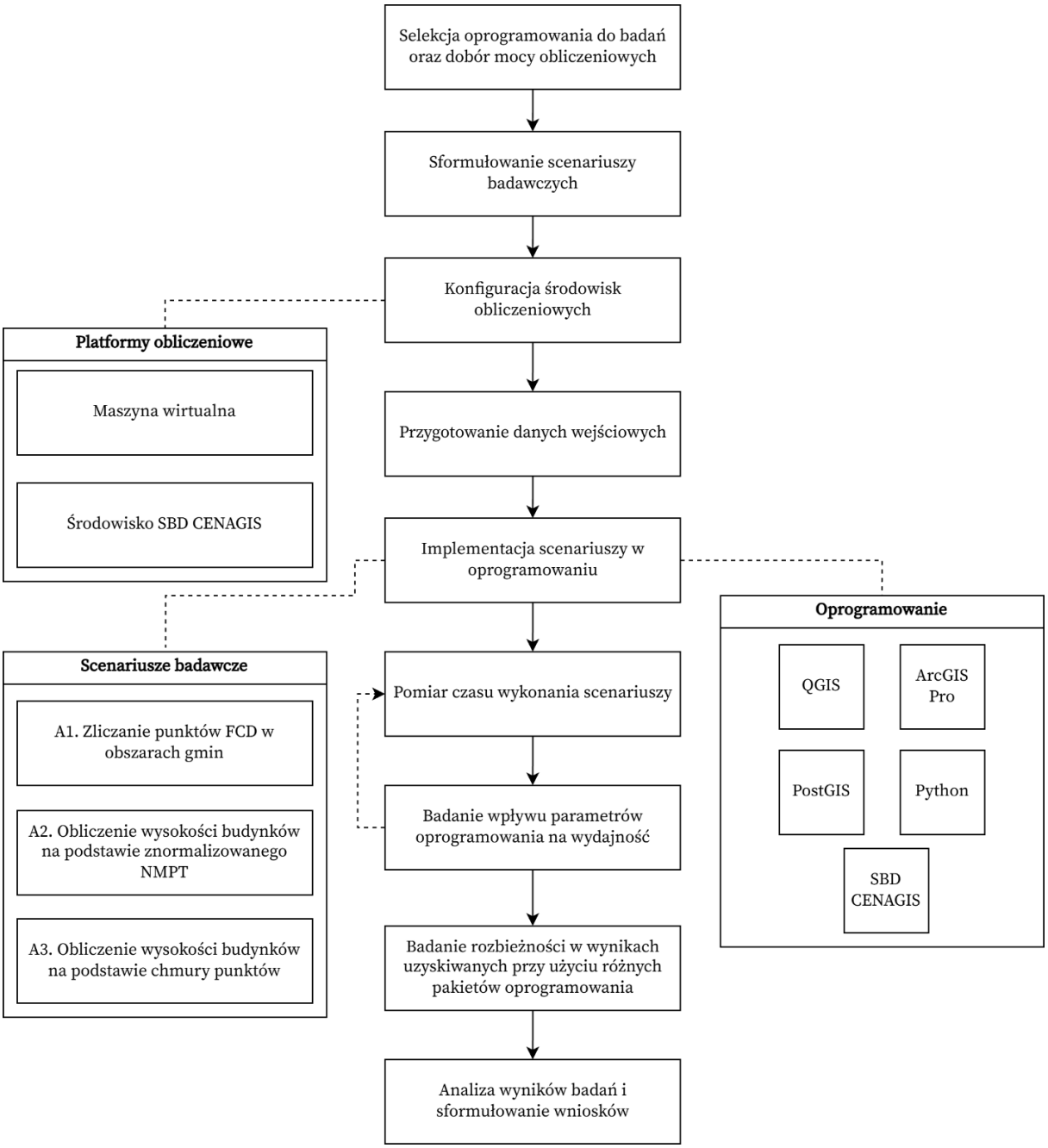
a) Obszaru podziału kraju w postaci plików ESRI SHAPEFILE:
    a. Obszary gmin wg. Państwowego Rejestru Granic (PRG)
    b. Statystyczna siatka kilometrowa zgodna z podziałem Głównego Urzędu Statystycznego (GUS)
    c. Heksagony H3[58]
        i. Rozdzielczość 8 (ok. 0.73 km2)
        ii. Rozdzielczość 9 (ok. 0.10 km2)
    d. Rejony statystyczne GUS
    e. Obwody spisowe GUS
b) Dane z Bazy Danych Obiektów Topograficznych (BDOT10k) pochodzącej z zasobów Głównego Urzędu Geodezji i Kartografii (GUGiK) w postaci plików ESRI SHAPEFILE:
    a. Warstwa BUBD – budynki
    b. Warstwa SKDR – drogi
    c. Warstwa BUIB – inne budowle
c) Modele terenu pochodzące z zasobów GUGiK w postaci plików GeoTiff:
    a. Numeryczny Model Terenu (NMT) o rozdzielczości 1 m
    b. Numeryczny Model Pokrycia Terenu (NMPT) o rozdzielczości 0.5 m
d) Chmury punktów w formacie LAZ pochodzące z zasobów GUGiK

Szczegółowe informacje dotyczące wykorzystania wskazanych powyżej zbiorów danych podano w opisie realizacji badań w Rozdziałach 9 oraz 10.

8.3. Metodyka badań porównawczych GIS i SBD

Badania porównawcze klasycznych analiz GIS oraz analiz zgodnych z metodyką SBD mają na celu weryfikacje możliwości i opłacalności wykorzystania rozwiązań SBD do realizacji analiz przestrzennych. Na poniższym diagramie przedstawiono ogólną metodykę tych badań.

[58] Hierarchiczny system reprezentacji powierzchni świata w formie heksagonalnej zaprojektowany przez firmę Uber. Składa się on z 16 poziomów rozdzielczości (od 0 do 15), które połączone są ze sobą hierarchią możliwą do odtworzenie poprzez indeks przestrzenny każdego z heksagonów lub pentagonów (Brodsky, 2018).



Rysunek 38. Schemat ilustrujący przebieg (metodykę) badań porównawczych GIS i SBD (opracowanie własne)

Badania rozpoczęto od selekcji oprogramowania oraz doboru mocy obliczeniowych do wykorzystania w eksperymentach. Przyjęte do analizy pakiety oprogramowania musiały reprezentować zarówno klasyczne narzędzia klasy desktop GIS, jak i środowiska programistyczne, aby uwzględnić jak największą liczbę przypadków zastosowania analiz GIS.

Następnie sformułowano scenariusze badawcze czyli konkretne analizy przestrzenne możliwe teoretycznie do wykonania z użyciem jednego komputera, ale stanowiące wyzwanie obliczeniowe w związku z wielkością/złożonością danych wejściowych. Analizowane

scenariusze oparto o trzy główne rodzaje danych przestrzennych: dane wektorowe, dane rastrowe oraz chmury punktów. Kolejnym krokiem była konfiguracja i zdefiniowanie parametrów platform obliczeniowych tj. środowiska SBD CENAGIS , w którym przeprowadzono badania z użyciem metod SBD i języka Python oraz maszyny wirtualnej, która posłużyła do badań z wykorzystaniem oprogramowania desktop oraz bazy danych PostGIS.

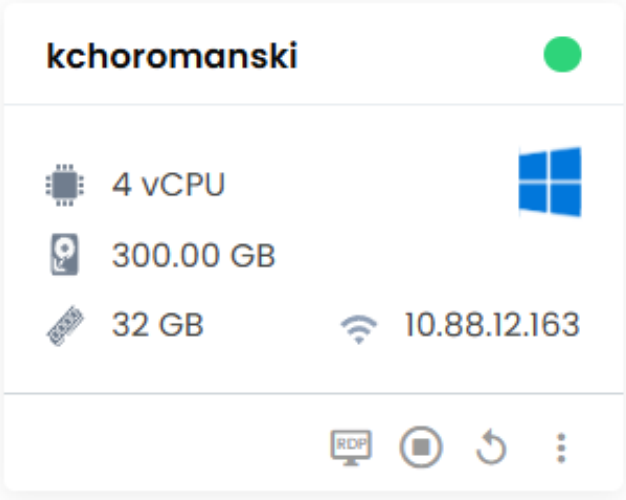
Implementacja scenariuszy badawczych obejmowała analizowane pakiety oprogramowania, przy czym ich wykorzystanie było determinowane specyfiką danego scenariusza. PostGIS został zastosowany wyłącznie w scenariuszu A1, dotyczącym analizy danych wektorowych z wykorzystaniem złączeń przestrzennych. Poza tym wyjątkiem wszystkie pozostałe scenariusze zostały zaimplementowane we wszystkich analizowanych pakietach oprogramowania. W trakcie realizacji scenariuszy badawczych dokumentowano oraz analizowano trudności implementacyjne wynikające ze specyfiki poszczególnych rozwiązań. W następnym kroku dokonywano pomiarów czasu wykonania wszystkich scenariuszy w każdym z pakietów oprogramowania. Zastosowano iteracyjny schemat pomiarów, w ramach którego systematycznie modyfikowano wskazane czynniki i analizowano ich wpływ na uzyskane wyniki. Pozwoliło to nie tylko na porównanie wydajności pomiędzy poszczególnymi pakietami oprogramowania, lecz także na identyfikację możliwych do osiągnięcia usprawnień wydajnościowych w obrębie każdego z analizowanych środowisk. Po zakończeniu pomiarów przeanalizowano również rozbieżności w produktach końcowych analiz, gdyż pomimo realizacji analogicznych etapów przetwarzania, różnice implementacyjne pomiędzy narzędziami mogą prowadzić do odmiennych wyników końcowych. Finalnie sformułowano wnioski z przeprowadzonych eksperymentów. W dalszej części tego rozdziału przedstawiono szczegóły doboru środowisk obliczeniowych, narzędzi GIS oraz scenariuszy badawczych analizowanych w tej części badań.

Do przeprowadzenia badań wyselekcjonowano pięć środowisk analizy GIS:

1) ESRI ArcGIS Pro 3.2.2,
2) QGIS 3.34.5,
3) PostGIS 3.4
4) Język programowania Python 3.7 wraz z bibliotekami analizy przestrzennej.
5) Środowisko *spatial big data* CENAGIS

Oprogramowanie ArcGIS jest obecnie najszerzej wykorzystywanym komercyjnym środowiskiem analiz GIS (Dempsey, 2019). QGIS stanowi natomiast najpopularniejszą alternatywę dostępną na zasadach *open source* (Dempsey, 2019). Baza danych przestrzennych PostGIS jest regularnie uwzględniana w dotychczasowych badaniach porównawczych zestawiających klasyczne narzędzia GIS z rozwiązaniami z obszaru *spatial big data* ((Huang, Chen, Wan i Peng, 2017), (Zhang i inni, 2022), (Haynes, Mitchell i Shook, 2020)). Z kolei język programowania Python został wskazany w badaniach ankietowych (patrz rozdział 6.2.2) jako najczęściej wykorzystywany język programowania w analizach danych przestrzennych.

Dobór mocy obliczeniowych dla przeprowadzenia tej części badań nie był oczywisty. Brak jest opublikowanych wyników badań dotyczących optymalnych mocy obliczeniowych do wykorzystania na potrzeby analiz desktop GIS. Oprogramowanie QGIS nie ma wskazanych rekomendowanych wymagań sprzętowych, a w przypadku oprogramowania ArcGIS Pro są to: 4 rdzenie CPU oraz 32 GB RAM[59]. W związku z powyższym analogiczne wartości przyjęto jako parametry maszyny wirtualnej utworzonej do badań opartych o narzędzia ArcGIS Pro, QGIS oraz PostGIS. Maszyna ta została skonfigurowana z systemem operacyjnym Windows Server 2019. Na rysunku 39 przedstawiono podsumowanie parametrów maszyny wirtualnej utworzonej do przeprowadzenia badań.



Rysunek 39. Podsumowanie parametrów maszyny wirtualnej wykorzystanej w badaniach porównawczych GIS i SBD do analiz z użyciem narzędzi Desktop GIS oraz bazy danych PostGIS widoczne na stronie panel.cenagis.edu.pl

[59] https://pro.arcgis.com/en/pro-app/latest/get-started/arcgis-pro-system-requirements.htm (data dostępu: 15.12.2025)

Analizy klasycznych narzędzi GIS oparto o technologię maszyn wirtualnych ze względu na fakt, iż do wykorzystania narzędzi Desktop GIS konieczny jest graficzny interfejs użytkownika. Wszystkie badane pakiety oprogramowania zainstalowano na maszynie wirtualnej w standardowej, domyślnej konfiguracji ustawień.

Badania realizowane z wykorzystaniem języka Python oraz jego bibliotek do przetwarzania danych przestrzennych przeprowadzono przy zastosowaniu tych samych parametrów mocy obliczeniowej, co w przypadku środowiska wirtualizacji, w celu zapewnienia porównywalności uzyskanych wyników. Analizy te wykonano jednak w ramach Podsystemu Big Data CENAGIS, z wykorzystaniem platformy Jupyter Notebook. Podsystem Big Data CENAGIS pozwala na dynamiczne tworzenie środowisk obliczeniowych w oparciu o technologię konteneryzacji.

Platforma obliczeniowa dla eksperymentów z użyciem języka Python posiadała następującą charakterystykę: 4 vCPU i 32 GB RAM. Środowisko było oparte o system operacyjny Ubuntu 18.04 LTS. Wersja języka Python była odpowiadająca tej, zastosowanej w środowisku SBD CENAGIS, w celu minimalizacji różnic wydajnościowych spowodowanych optymalizacjami w poszczególnych pakietach oprogramowania.

Ze względu na specyfikę obliczeń realizowanych przez Apache Spark nie jest możliwe bezpośrednie porównanie wydajności klastra Spark do wydajności innych narzędzi GIS, gdyż analizy rozproszone realizowane są przez wiele jednostek obliczeniowych na raz. Obliczenia Spark obciążone są jednak narzutem związanym m.in. z transferem danych oraz inicjalizacją środowiska. W celu zbadania różnic wydajności przy zbliżonych mocach obliczeniowych z wykorzystaniem SBD oraz innych narzędzi, a także zweryfikowania zmiany wydajności w przypadku skalowania środowiska obliczeniowego przeprowadzono badania w dwóch konfiguracjach klastra:

- *Driver*: 4 vCPU, 32GB RAM, *Worker*: 1 x (4 vCPU, 32 GB RAM)
- *Driver*: 4 vCPU, 32 GB RAM, *Worker*: 50 x (4 vCPU, 32 GB RAM)

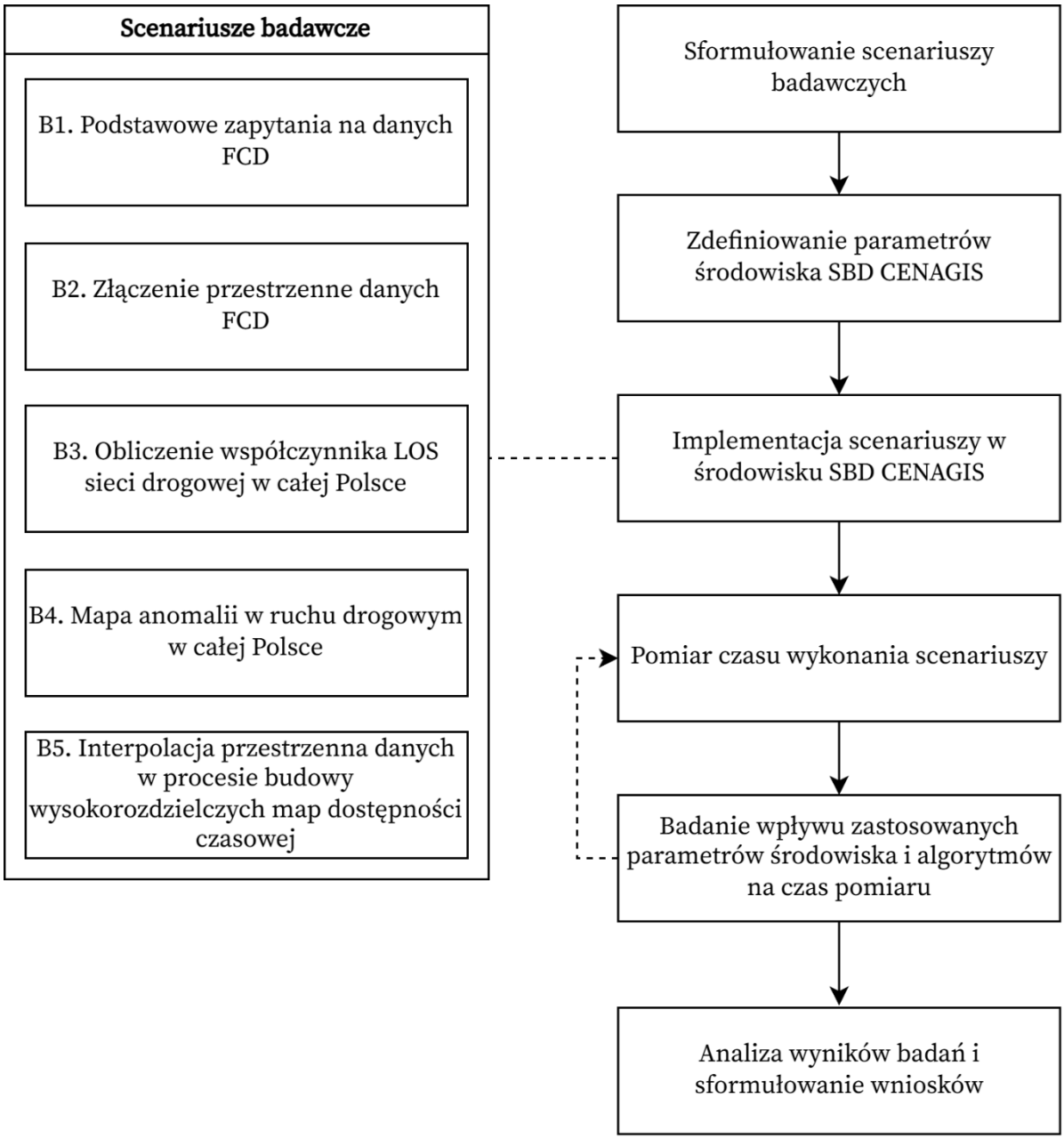
Kluczowym aspektem badań porównawczych na tym etapie było zdefiniowanie scenariuszy badawczych, umożliwiających możliwie kompleksową analizę wydajności procesów realizacji analiz geoprzestrzennych, przy jednoczesnym uwzględnieniu przyjętych ograniczeń i założeń badawczych.

W celu zapewnienia jak najlepszej porównywalności wyników założono, że scenariusze badawcze muszą opierać się na stosunkowo niezłożonych, a jednocześnie często wykorzystywanych w praktyce operacjach przestrzennych. Należy przy tym wyraźnie podkreślić, że głównym celem badania jest sprawdzenie wydajności poszczególnych metod w kontekście wykorzystania szeroko stosowanych operacji przestrzennych, a nie maksymalizacja wartości merytorycznej wyników konkretnych analiz. W związku z tym w przyjętych scenariuszach zrezygnowano z wykonywania dodatkowych operacji takich jak filtracja pomiarów odstających, redukcja szumów czy analiza przypadków szczególnych. Szczegółowe opisy wszystkich zrealizowanych w ramach badań scenariuszy przedstawiono w rozdziale 9.

W tej części badań postawiono następującą hipotezę badawczą: „Nie dla wszystkich analiz przestrzennych konieczne i uzasadnione jest wykorzystanie narzędzi i technologii SBD."

8.4. Metodyka badań wydajności środowiska SBD

Badania wydajności środowiska SBD mają na celu weryfikacje możliwości oraz wydajności przeprowadzania analiz na bardzo dużych zbiorach danych z użyciem badanego środowiska SBD – Podsystemu Big Data CENAGIS. Badanie wydajności powinno odbyć się z uwzględnieniem wielkości danych wejściowych, ich rodzaju oraz zastosowanych optymalizacji obliczeń. Podobnie jak w przypadku badań porównawczych omówionych w rozdziale 8.3, eksperymenty na tym etapie oparto o scenariusze badawcze. Na poniższy diagramie zaprezentowano metodykę przyjętą w tej części badań.



Rysunek 40. Schemat ilustrujący przebieg badań (metodykę) wydajności metod SBD (opracowanie własne)

Przyjęto następujące zasady podstawowe dla opracowania scenariuszy badawczych:

- Realizacja zagadnień tematycznych stanowiących częsty i istotny temat analiz z użyciem bardzo dużych zbiorów danych przestrzennych (np. transport, ekologia, epidemiologia, bezpieczeństwo, urbanistyka – obszary wskazane w licznych pracach z zakresu SBD m.in. (Wu, Gan, Chao i Yu, 2024) i (Dritsas i Trigka, 2025))
- Rozmiar danych powinien uniemożliwiać lub bardzo utrudniać przeprowadzenie pełnej analizy z użyciem systemów GIS działających na jednym komputerze
- Scenariusze powinny testować możliwości i wydajność analiz zarówno danych wektorowych, jak i rastrowych

- W ramach realizacji scenariuszy badaniom powinna podlegać:
    - Wydajność (szybkość obliczeń)
    - Stabilność (podatność na błędy, powtarzalność wyników)
    - Trudność implementacji (konieczność wprowadzania optymalizacji i wdrażania brakujących funkcji obliczeniowych, trudność wdrożenia założonych scenariuszy w porównaniu do podejść klasycznych)

W celu utrzymania możliwie wielu podobieństw pomiędzy badaniami porównawczymi przedstawionymi w rozdziale 8.3, w każdym z badanych scenariuszy maszyna *driver* posiadała parametry identyczne do maszyny wykorzystywanej w ramach badań porównawczych tj. 4 vCPU, 32 GB RAM. Ze względu na specyfikę poszczególnych scenariuszy badań wydajnościowych nie było jednak możliwe zastosowanie w każdym z nich tych samych parametrów klastra. W ramach jednego ze scenariuszy (scenariusz B5) zbadano jednak wpływ dobranych parametrów na wydajność przetworzeń. Konieczność zastosowania różnych parametrów powodowana była wymaganiami co do ilości pamięci operacyjnej w poszczególnych scenariuszach, obciążeniem klastra, a także ograniczonym czasem, w jakim możliwe było przeprowadzenie obliczeń z zachowaniem wszystkich warunków co do poprawności pomiaru. W załączniku 1. przedstawiono wszystkie pozostałe zmienne konfiguracyjne środowiska Spark zastosowane do przeprowadzenia badań. Szczegółowe opisy wszystkich zrealizowanych w ramach badań scenariuszy przedstawiono w rozdziale 9.

W tej części badań postawiono następującą hipotezę badawczą: „Zastosowanie technik obliczeń rozproszonych w środowisku *spatial big data* opartym o Apache Spark umożliwia wydajne przeprowadzenie analiz przestrzennych bardzo dużych zbiorów danych o mobilności."

8.5. Metodyka pomiarów

Przyjęto następujące założenia dotyczące metodyki pomiarów czasu wykonania poszczególnych scenariuszy badawczych, realizowanych w obu etapach badań:

- Każde badanie należy powtórzyć przynajmniej 5 razy, zwiększając tę liczbę do 10 prób w przypadku, gdy czas obliczeń na to pozwala. Przyjęto, że akceptowalny czas wykonania pojedynczego pomiaru, umożliwiający realizację pełnej serii dziesięciu powtórzeń, nie powinien przekraczać dwóch godzin. W celu ograniczenia wpływu mechanizmów buforowania na wyniki pomiarów nie uwzględniano pierwszego

uruchomienia w danej serii w analizie porównawczej, jednak pomiar ten był każdorazowo rejestrowany i analizowany pod kątem istotnych odchyleń. W przypadku stwierdzenia znaczących różnic pomiędzy pierwszym a kolejnymi uruchomieniami podejmowano działania mające na celu ograniczenie wpływu agresywnego buforowania danych prowadzącego do sztucznego przyspieszenia obliczeń w kolejnych iteracjach. Ze względu na złożoność obliczeń rozproszonych należy dążyć do minimalizacji wpływu czynników zewnętrznych (obciążenie klastra, problemy w komunikacji sieciowej) na wynik. Pomiary istotnie odbiegające od średniej należy przeanalizować pod względem wpływu anomalii zewnętrznych.

- W celu ograniczenia czynników zewnętrznych obliczenia mogą być wykonywane przy maksymalnym obciążeniu klastra wynoszącego nie więcej niż 20% zarówno pod względem liczby vCPU, jak i ilości pamięci RAM.
- W przypadku narzędzi opartych na Apache Spark metodyka pomiaru uwzględnia mechanizm tzw. „ewaluacji leniwej", w którym operacje są wykonywane dopiero w momencie prezentacji wyników. Aby uniknąć zafałszowania wyników wynikającego z działania tego mechanizmu w Apache Spark, czas mierzono dopiero po wymuszeniu wykonania obliczeń poprzez akcję, tj. zapis wyniku na HDFS lub jego pobranie do maszyny *driver*. Dzięki temu pomiar obejmował rzeczywisty czas wykonania transformacji. Przed wykonaniem pomiarów należy dokonać weryfikacji poprawności działania klastra poprzez sprawdzenie działania wszystkich usług (Mesos, HDFS, marathon-lb), w szczególności pod względem występowania problemów sieciowych bądź z autoryzacją użytkownika.
- W trakcie trwania obliczeń należy weryfikować poprawne przypisanie wszystkich zadeklarowanych mocy obliczeniowych do analizy (błąd w trakcie obliczeń bądź problem z jednym z serwerów może spowodować nieprzypisanie całych zadeklarowanych mocy pomimo ich teoretycznej dostępności).

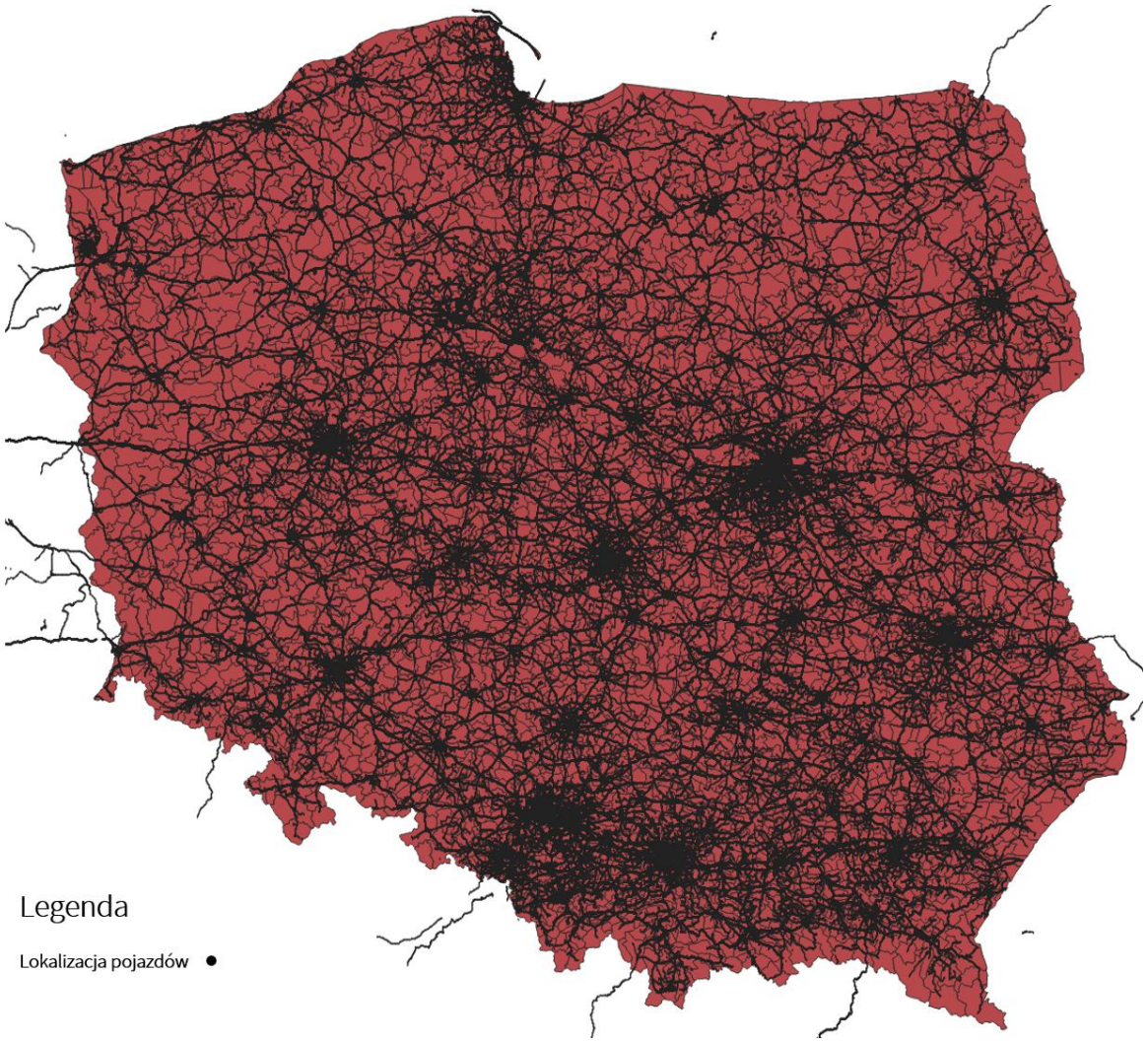
9. Badania porównawcze narzędzi GIS i SBD

W tym rozdziale przedstawione zostały scenariusze badań porównawczych oraz wyniki ich realizacji zgodnie z metodyką przyjętą w rozdziale 8.3.

9.1. Scenariusz A1 – zliczanie punktów FCD w obszarach gmin

Scenariusz A1 polega na realizacji funkcji zliczania rekordów z danych punktowych FCD z aplikacji Yanosik w gminach w Polsce dla jednego dnia pomiaru tj. 11.04.2020. Liczba rekordów dla tego dnia pomiaru wyniosła 36.6 miliona, co przekłada się na 2.5 GB danych w postaci pliku CSV. Z kolei dane o gminach z Państwowego Rejestru Granic (PRG) zawierają 2477 rekordów, co odpowiada 80 MB pliku ESRI SHAPEFILE. Operacja zliczenia obiektów w jednostkach powierzchniowych należy do jednej z najbardziej podstawowych operacji przestrzennych. Wymaga ona wykonania operacji złączenia przestrzennego, która to jest częstym przedmiotem badań wydajności SBD (Huang, Chen, Wan i Peng, 2017) i (Kim, Hoang, Yu i Kanwar, 2023). Założono, że złączenie przestrzenne w tym scenariuszu zostanie wykonane z użyciem operatora *Contains* zgodnie z modelem relacji topologicznych DE-9IM (ang. *Dimensionally Extended Nine-Intersection Model*) (Clementini i Di Felice, 1996)

Na rysunku 41 przedstawiono uproszczoną wizualizację danych punktowych FCD na tle obszaru Polski.



Rysunek 41. Wizualizacja danych o lokalizacjach pojazdów z aplikacji Yanosik dla jednego dnia pomiarów (11.04.2020) na tle Polski. Punkty pomiarowe oznaczono kolorem czarnym (opracowanie własne)

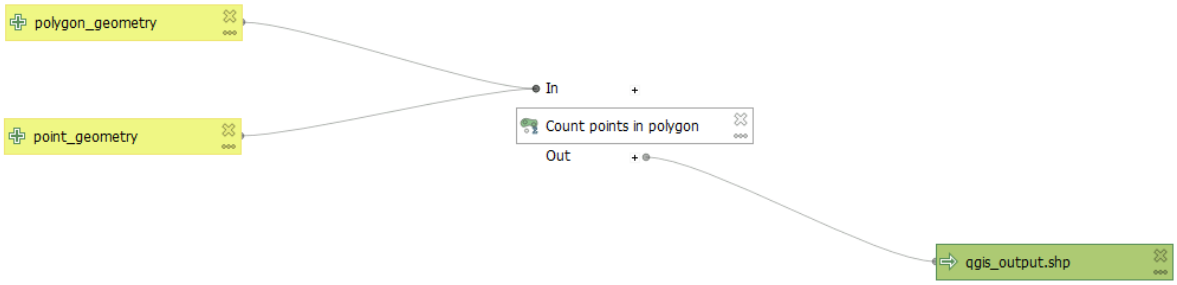
Scenariusz A1 został zrealizowany zgodnie z przyjętymi wcześniej założeniami w 5. badanych pakietach oprogramowania: QGIS, ArcGIS Pro, PostGIS, Python oraz SBD CENAGIS.

Pierwszym badanym środowiskiem w scenariuszu A1 było oprogramowanie QGIS. Realizacja tego zadania w oprogramowaniu QGIS sprowadza się do trzech prostych kroków:

- Wczytanie obu zbiorów danych (obszary gmin oraz warstwa z punktami pomiarowymi FCD)
- Wywołanie funkcji „*Count points in polygon*” (zliczanie punktów w obszarze poligonów)
- Zapis wynikowego zbioru danych do pliku

W celu minimalizacji niezbędnych kroków realizacji obliczeń zrezygnowano z wizualizacji zbioru danych na podkładzie mapowym. Wykorzystano narzędzie „Model Designer” do zdefiniowania

procedury obliczeń. Na rysunku poniżej zaprezentowano schemat przetworzeń z oprogramowania Model Designer.



Rysunek 42. Schemat blokowy z modułu Model Designer oprogramowania QGIS utworzony do realizacji scenariusza A1 (opracowanie własne)

Zmierzono zarówno czas wczytania danych z pamięci masowej do pamięci operacyjnej, a także czas wykonania algorytmu i zapisu wyników. W przypadku analizy z wykorzystaniem pliku CSV model został rozbudowany o dodatkowy moduł interpretujący odpowiednie kolumny zawierające informacje o geometrii w pliku CSV. Ze względu na fakt, że zwykle bardzo istotny wpływ na szybkość wykonywanych przetworzeń ma format analizowanych danych przestrzennych oraz zastosowana metoda indeksowania przestrzennego, zbadano czasy odczytu pliku zarówno z jak i bez utworzonego indeksu przestrzennego.

Zgodnie z oczekiwaniami zastosowanie indeksowania przestrzennego znacząco przyspieszyło obliczenia w QGIS. W przypadku pliku CSV indeks przestrzenny w QGIS nie zostanie wygenerowany pomimo zastosowania narzędzia „*Create Spatial Index*”, co może prowadzić do złej interpretacji problemów z przetwarzaniem dużych zbiorów danych przestrzennych w tym oprogramowaniu. Spowodowane jest to specyfiką tego narzędzia, które generuje indeks przestrzenny tylko w przypadku, gdy wejściowy plik z danymi wspiera taką możliwość.

Czas tworzenia indeksu przestrzennego w przypadku pliku SHAPEFILE zajął średnio 147 sekund. Po transformacji pliku wejściowego do postaci SHAPEFILE średni czas obliczeń skrócił się o 1 godzinę 16 minut, a zastosowanie dodatkowo indeksowania przestrzennego skróciło ten czas z ok. 8g. 30 min. do ok. 39 minut. Porównanie czasów wykonania w różnych wariantach przedstawiono w tabeli 10Tabela 10.

Tabela 10. Porównanie czasów zliczenia punktów wewnątrz poligonów w QGIS dla wybranych formatów z i bez zastosowania indeksowania przestrzennego (opracowanie własne)

Próba	CSV - bez indeksu przestrzennego [gg:mm:ss]	SHAPEFILE - z indeksem przestrzennym [gg:mm:ss]	SHAPEFILE - bez indeksu przestrzennego [gg:mm:ss]
1	09:50:12	00:37:41	08:36:30
2	09:49:59	00:38:50	08:17:13
3	09:47:20	00:39:09	08:45:09
4	09:52:39	00:38:26	08:27:32
5	09:45:48	00:39:02	08:38:28
6	09:45:48	00:39:02	08:38:28
7	09:47:58	00:39:12	08:28:02
8	09:50:09	00:38:42	08:45:46
9	09:46:55	00:39:17	08:32:29
10	09:48:10	00:38:54	08:19:45
Średnia	**09:48:38**	**00:38:46**	**08:33:04**

Najdłuższym procesem przetworzeń było wykonanie operacji zliczania punktów w poligonach, Dla procesu z użyciem formatu SHAPEFILE z indeksem przestrzennym (najbardziej wydajna opcja) ładowanie danych zajęło średnio 4 minuty 37 sekundy, podczas gdy operacja zliczania wymagała średnio 38. minut 46. sekund. Zestawianie czasów obliczeń w rozbiciu na etapy zaprezentowano w tabeli 11.

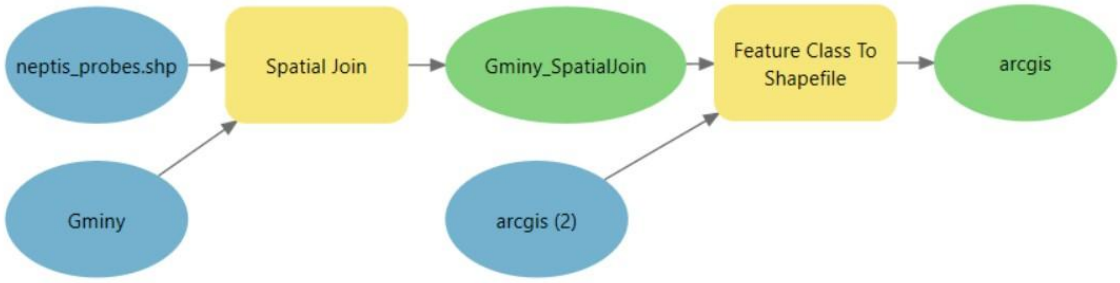
Tabela 11. Czas trwania poszczególnych procesów w ramach najszybszej badanej metody - SHAPEFILE z indeksem przestrzennym – scenariusz A1, oprogramowania QGIS (opracowanie własne)

Próba	Ładowanie danych [gg:mm:ss]	Złączenie przestrzenne i agregacja [gg:mm:ss]	Całość [gg:mm:ss]
1	00:04:30	00:33:11	00:37:41
2	00:04:34	00:34:16	00:38:50
3	00:04:38	00:34:31	00:39:09
4	00:04:32	00:33:55	00:38:26
5	00:04:49	00:34:13	00:39:02
6	00:05:01	00:34:11	08:38:28
7	00:04:44	00:33:58	08:28:02
8	00:05:11	00:34:06	08:45:46
9	00:04:25	00:34:29	08:32:29
10	00:04:55	00:33:36	08:19:45
Średnia	**00:04:44**	**00:34:02**	**00:38:46**

W oprogramowaniu ArcGIS Pro zrealizowano analizę z użyciem narzędzia ModelBuilder. Poniżej przedstawiono zrealizowane kroki:

- Wczytanie danych FCD oraz obszarów gmin
- Wykonanie złączenia przestrzennego funkcją *Spatial Join*. Wykorzystano sposób złączenia „Całkowicie zawiera” (*Completely Within*)
- Zapis wyniku do pliku SHP na dysku

Na rysunku 43 przedstawiono schemat blokowy z modułu ModelBuilder.



Rysunek 43. Schemat blokowy z modułu ModelBuilder oprogramowania ArcGIS Pro utworzony do realizacji scenariusza A1 (opracowanie własne)

Należy zauważyć, że w przypadku oprogramowania ArcGIS Pro nie można wykonać tej operacji bezpośrednio z pliku tekstowego CSV (musi i tak być on wcześniej przetworzony do formy warstwy przestrzennej). Nie przeanalizowano więc różnic wynikających z wejściowej formy danych.

W 10. próbach czas przetworzeń z użyciem oprogramowania ArcGIS Pro wyniósł średnio 31 minut 30 sekund (szczegóły pomiarów przedstawiono w tabeli 12).

Tabela 12. Podsumowanie pomiarów czasu wykonania scenariusza A1 z użyciem oprogramowania ArcGIS Pro (opracowanie własne)

Próba	Odczyt i złączenie przestrzenne [gg:mm:ss]	Zapis do SHP [gg:mm:ss]	Całość [gg:mm:ss]
1	00:30:36	00:00:05	00:30:41
2	00:31:49	00:00:05	00:31:54
3	00:31:26	00:00:04	00:31:30
4	00:31:41	00:00:05	00:31:46
5	00:31:26	00:00:06	00:31:32
6	00:30:48	00:00:07	00:30:55
7	00:32:03	00:00:05	00:32:08
8	00:31:44	00:00:05	00:31:49
9	00:31:34	00:00:04	00:31:38
10	00:30:58	00:00:06	00:31:04
Średnia	**00:31:24**	**00:00:05**	**00:31:30**

Jest to czas zbliżony do tego oferowanego przez oprogramowanie QGIS z zastosowaniem analogicznych danych wejściowych, jednak zauważyć można w tym przypadku regularną, wyższą wydajność oprogramowania ArcGIS (różnica wynosi średnio 7 minut na korzyść ArcGIS). Poziom trudności implementacji scenariusza w obu najpopularniejszych systemach GIS jest bardzo zbliżony. Jednak w przypadku ArcGIS istnieje mniejsze ryzyko popełnienia błędów znacznie wpływających na wydajność przetwarzania takich jak nieutworzenie indeksu przestrzennego.

Trzecim analizowanym oprogramowaniem w ramach przetwarzania danych wektorowych była baza danych przestrzennych PostGIS. Baza PostGIS została w tym przypadku wykorzystana w roli narzędzia GIS. Została więc zainstalowana na maszynie testowej, z której następnie uruchomiono odpowiednie zapytania. W przypadku bazy danych PostGIS w pierwszym kroku zaimportowano dane w postaci pliku SHAPEFILE do tabeli z wykorzystaniem środowiska QGIS. Dane o obszarach gmin zaimportowano do tabeli „Gminy", a dane punktowe z aplikacji Yanosik do tabeli „*neptis_probes*". Następnie założono indeks przestrzenny oparty o algorytm R-Drzewa (Guttman, 1984) na obu tabelach. Finalnie wykonano kwerendę zliczającą wystąpienie punktów wewnątrz poligonów kwerendą przedstawioną na rysunku 44).

```
SELECT
  g.jpt_kod_je AS jpt_kod_je,
  COUNT(p.id) AS point_count
FROM
  "Gminy" g
LEFT JOIN
  neptis_probes p
ON
        ST_Within(p.geom, g.geom)
GROUP BY
  g.jpt_kod_je;
```

Rysunek 44. Kwerenda wykorzystana do realizacji scenariusza A1 z użyciem bazy danych PostreSQL z rozszerzeniem PostGIS (opracowanie własne)

Plan wykonania kwerendy (rysunek 45) zakładał lewostronne złączenie informacji o gminach do punktów pomiarowych, a następnie agregacje tych danych poprzez liczenie wystąpień dla poszczególnych gmin. Plan ujawnia wykorzystanie indeksowania przestrzennego (operator @ poszukiwania wewnątrz prostokąta ograniczającego).

#	Node
1.	→ Aggregate
2.	→ Nested Loop Left Join
3.	→ Seq Scan on Gminy as g
4.	→ Index Scan using sidx_neptis_probes_geom on neptis_probes as p **Filter**: st_within(geom, g.geom) **Index Cond**: (geom @ g.geom)

Rysunek 45. Plan wykonania kwerendy realizującej scenariusz A1 z użyciem bazy danych PostGIS pochodzący z oprogramowania PGAdmin (opracowanie własne)

Realizacja scenariusza z wykorzystaniem PostGIS trwała średnio 5 minut 25 sekund w 10. próbach. Należy zwrócić uwagę na wysoką stabilność obliczeń w przypadku oprogramowania PostGIS (odchylenie standardowe z pomiarów na poziomie poniżej 6 sekund). Wszystkie pomiary przedstawiono w tabeli 13.

Tabela 13. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A1 z wykorzystaniem bazy danych PostGIS (opracowanie własne)

Próba	PostGIS [gg:mm:ss]
1	00:05:26
2	00:05:16
3	00:05:29
4	00:05:31
5	00:05:31
6	00:05:18
7	00:05:28
8	00:05:34
9	00:05:19
10	00:05:22
Średnia	00:05:25

W przypadku oprogramowania PostGIS w czas pomiaru nie włączono czasu zapisu wyników do pliku, gdyż pomiarowi podlegało wykonanie kwerendy, której wynik następnie zapisywano do pliku tekstowego. Sama operacja zapisu danych wynikowych do pliku zajęła średnio 5 sekund z wykorzystaniem oprogramowania PGAdmin.

Kolejnym narzędziem wykorzystanym w porównaniu był język programowania Python. Wykorzystano bibliotekę geopandas do realizacji scenariusza opartego o dane wektorowe. W ramach realizacji zadania przygotowano skrypt realizujący następującą listę kroków:

- Wczytanie danych o pomiarach FCD (plik CSV) oraz gminach (plik SHAPEFILE)
- Transformacja układu współrzędnych gmin do układu danych punktowych (EPSG:4326)

- Podział danych FCD na podzbiory po 100 tysięcy rekordów
- Wykonanie złączenia przestrzennego z warstwą gmin dla wszystkich podzbiorów danych FCD
- Zsumowanie wystąpień wszystkich gmin we wszystkich podzbiorach
- Dołączenie uzyskanej liczby do warstwy geometrycznej gmin

Takie podejście porównano do analogicznego podejścia bez podziału danych na podzbiory. W takim wypadku operacja konsumuje całą dostępną pamięć RAM, co może mieć wpływ na wydajność obliczeń.

Implementacja tego scenariusza w języku Python była stosunkowo prosta. Kod pozbawiony dodatkowych komentarzy i elementów służących do pomiaru czasu wykonania zajmuje 26 linijek w wersji z podziałem na podzbiory. Jednocześnie implementacja z użyciem języka programowania pozwala na dużo większą kontrolę nad poszczególnymi etapami przetwarzania w porównaniu do oprogramowania GIS. Umożliwiło to bardziej szczegółowy pomiar poszczególnych etapów realizacji analizy.

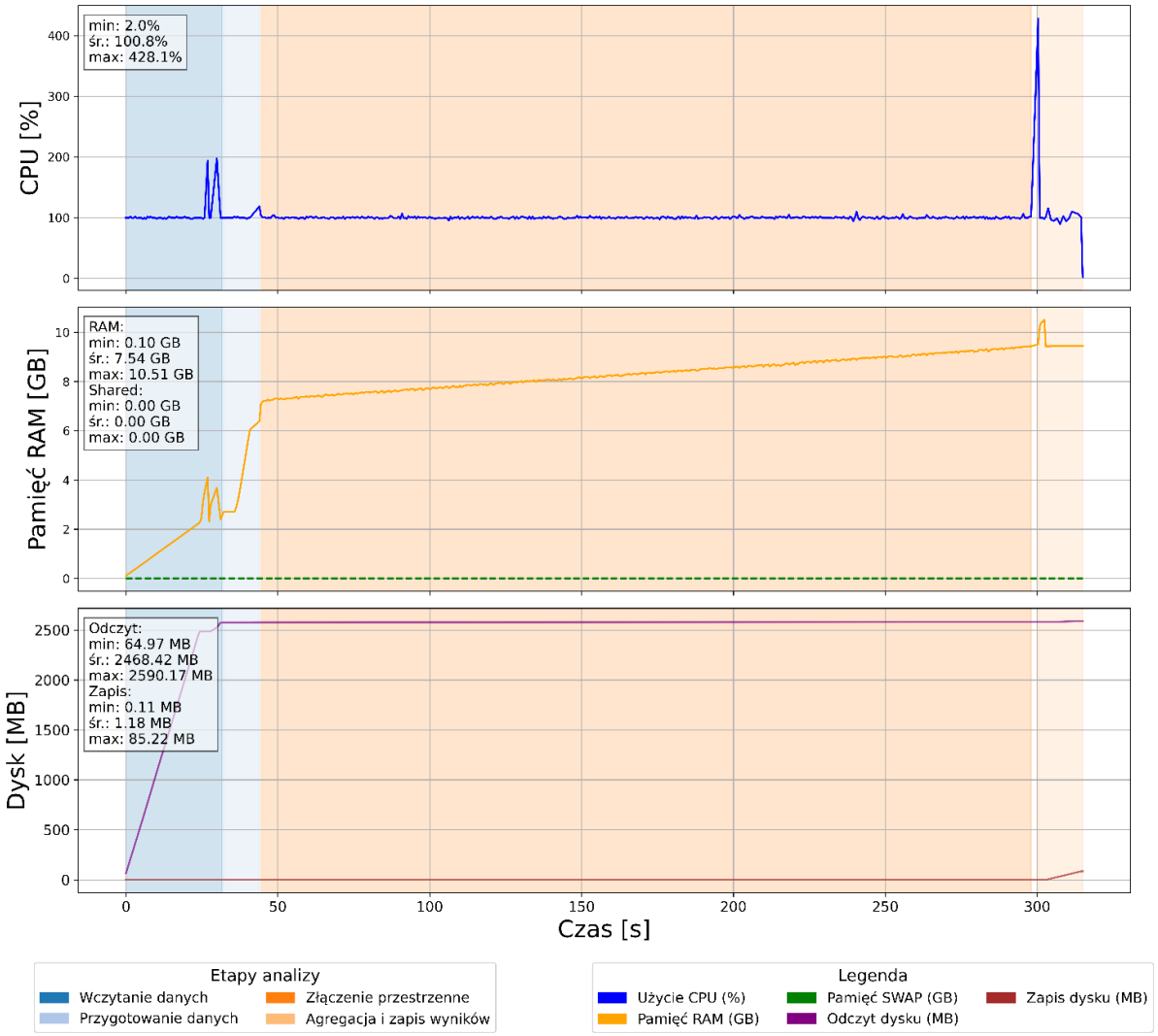
Średni czas przetwarzania przy zastosowaniu optymalizacji polegającej na rozbiciu obliczeń na podzbiory wyniósł 5 minut 21 sekund czyli o ponad 30 sekund szybciej, niż w przypadku braku rozbicia (szczegóły przedstawiono w tabeli 14). Powodem dłuższego czasu obliczeń w przypadku braku rozbicia jest najprawdopodobniej konsumpcja całej pamięci RAM w trakcie tego procesu. Przed rozpoczęciem obliczeń zajętość pamięci RAM wyniosła 812 MB, a w trakcie tego procesu cały zasób został wysycony powodując zapis do pamięci *swap*[60].

[60] Pamięć *swap* to obszar na dysku twardym wykorzystywany przez system operacyjny jako rozszerzenie pamięci RAM. Gdy fizyczna pamięć operacyjna jest niewystarczająca, rzadziej używane dane są tymczasowo przenoszone do pamięci swap, co pozwala uniknąć przerwania działania procesów kosztem znacznego spadku wydajności.

Tabela 14. Wyniki pomiarów czasu realizacji scenariusza A1 z użyciem języka programowania Python w rozbiciu na poszczególne etapy przetwarzania w dwóch wariantach - z i bez podziału na podzbiory (opracowanie własne)

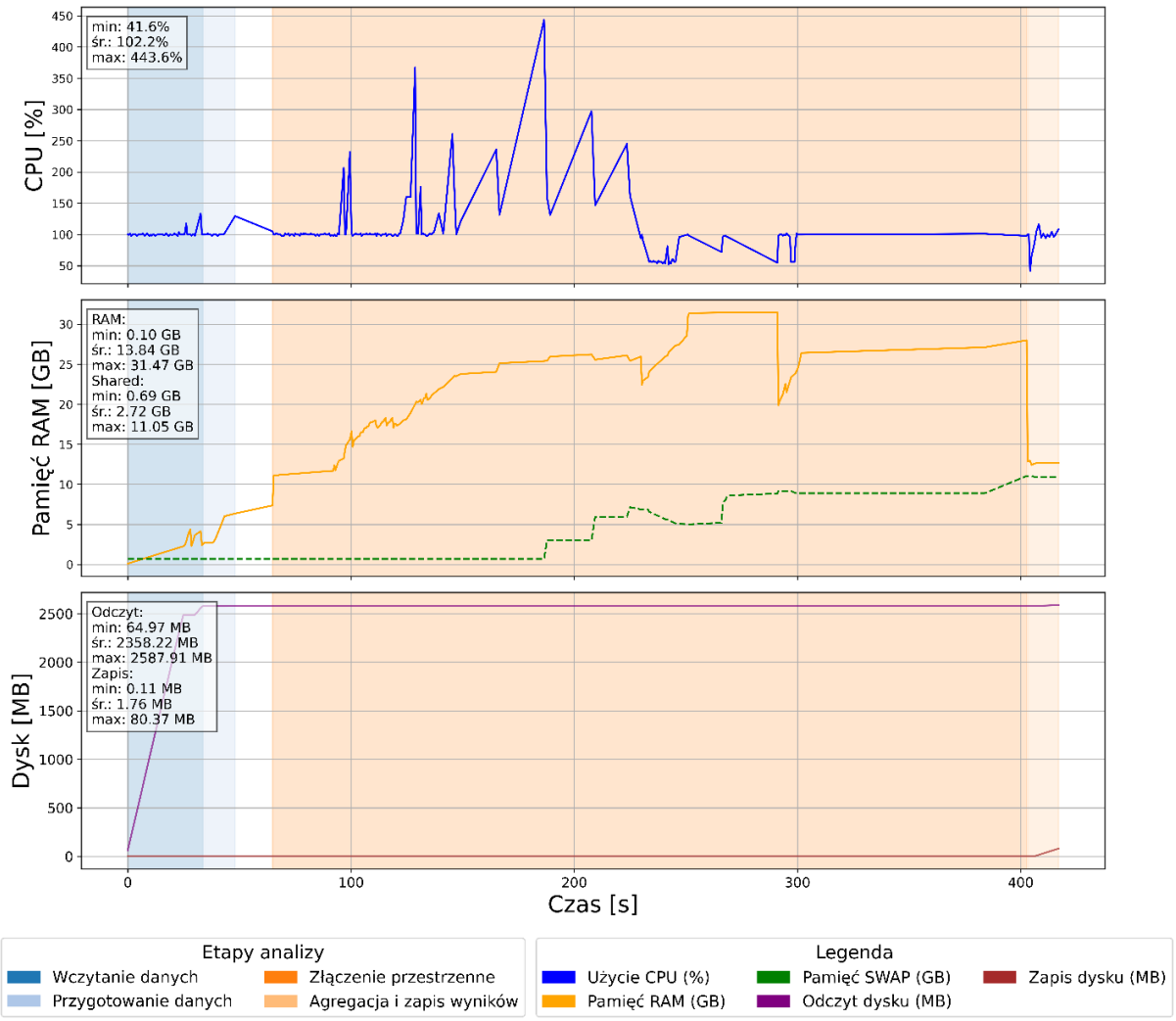
Podział na podzbiory						
Próba	Ładowanie danych [gg:mm:ss]	Przygotowanie danych [gg:mm:ss]	Złączenie przestrzenne [gg:mm:ss]	Agregacja [gg:mm:ss]	Zapis danych [gg:mm:ss]	Suma [gg:mm:ss]
1	00:00:32	00:00:13	00:04:27	00:00:04	00:00:13	00:05:29
2	00:00:32	00:00:11	00:04:21	00:00:05	00:00:13	00:05:21
3	00:00:32	00:00:11	00:04:20	00:00:04	00:00:12	00:05:18
4	00:00:32	00:00:11	00:04:23	00:00:05	00:00:12	00:05:23
5	00:00:32	00:00:13	00:04:22	00:00:05	00:00:12	00:05:24
6	00:00:31	00:00:11	00:04:20	00:00:06	00:00:13	00:05:21
7	00:00:33	00:00:12	00:04:19	00:00:05	00:00:12	00:05:22
8	00:00:32	00:00:11	00:04:21	00:00:06	00:00:12	00:05:22
9	00:00:32	00:00:12	00:04:18	00:00:04	00:00:12	00:05:18
10	00:00:33	00:00:12	00:04:18	00:00:05	00:00:12	00:05:21
Średnia	**00:00:32**	**00:00:12**	**00:04:21**	**00:00:05**	**00:00:12**	**00:05:22**
Brak podziału na podzbiory						
Próba	Ładowanie danych [gg:mm:ss]	Przygotowanie danych [gg:mm:ss]	Złączenie przestrzenne [gg:mm:ss]	Agregacja [gg:mm:ss]	Zapis danych [gg:mm:ss]	Suma [gg:mm:ss]
1	00:00:32	00:00:13	00:05:42	00:00:03	00:00:13	00:06:44
2	00:00:29	00:00:11	00:05:01	00:00:05	00:00:14	00:05:60
3	00:00:39	00:00:11	00:04:44	00:00:06	00:00:13	00:05:52
4	00:00:33	00:00:10	00:05:06	00:00:06	00:00:14	00:06:09
5	00:00:34	00:00:10	00:04:34	00:00:05	00:00:13	00:05:37
6	00:00:33	00:00:11	00:04:36	00:00:06	00:00:13	00:05:39
7	00:00:33	00:00:10	00:05:09	00:00:05	00:00:12	00:06:09
8	00:00:32	00:00:10	00:04:33	00:00:06	00:00:14	00:05:35
9	00:00:33	00:00:10	00:04:27	00:00:05	00:00:13	00:05:28
10	00:00:29	00:00:10	00:04:51	00:00:05	00:00:14	00:05:49
Średnia	**00:00:33**	**00:00:11**	**00:04:52**	**00:00:05**	**00:00:13**	**00:05:54**

Analiza zużycia zasobów (rysunek 46) w poszczególnych etapach analizy wskazuje, że niemal cały proces, z wyłączeniem etapu agregacji danych, został przeprowadzony jednowątkowo. Plik CSV zajmujący 2.4 GB pamięci na dysku przełożył się na maksymalne zużycie pamięci RAM w okolicach 10.51 GB. Odczyt danych w całości odbył się w pierwszym etapie analizy, jednak dopiero transformacja danych do formy obiektu biblioteki GeoPandas spowodowała wysokie wykorzystanie pamięci RAM (na poziomie ok. 6.5 GB), które wzrosło do okolic 10GB liniowo w trakcie wykonywania operacji złączenia przestrzennego.



Rysunek 46. Wykres wykorzystania poszczególnych zasobów (CPU, RAM, Dysk) w czasie trwania obliczeń dla scenariusza A1 w języku programowania Python w wariancie z rozbiciem zbioru (opracowanie własne)

W przypadku, gdy nie zastosowano dzielenia zbioru w trakcie operacji złączenia zaobserwować można wysycenie całej dostępnej pamięci RAM i konieczność składowania wyników w pamięci *swap*, co przełożyło się na dłuższy czas obliczeń (rysunek 47).



Rysunek 47. Wykres wykorzystania poszczególnych zasobów (CPU, RAM, Dysk) w czasie trwania obliczeń dla scenariusza A1 w języku programowania Python w wariancie bez rozbicia zbioru na części (opracowanie własne)

Realizacja scenariusza w oparciu o Apache Spark wymagała bardziej skomplikowanych operacji, w porównaniu do poprzednich narzędzi. Pierwszym krokiem była transformacja danych z oryginalnego formatu do formatu Parquet GeoMesa-FS i zapis go na HDFS. Wykonanie przetworzeń bezpośrednio z pliku CSV składowanego lokalnie (na maszynie *driver*) jest możliwe, ale powoduje dodatkowy narzut czasowy związany z przesłaniem zbioru do wszystkich maszyn obliczeniowych. Sama ta operacja zajęła 4min 30s (pomiar uśredniono w 10 próbach). Proces zapisu danych w postaci GeoMesa-FS wymusza również transformacje do układu EPSG:4326.

Czas złączenia przestrzennego zbadano z wykorzystaniem metody opartej o złączenie geopandas uruchomione w postaci funkcji pandas UDF (metoda opisana szerzej w rozdziale 10.2). Przetwarzanie danych z użyciem Apache Spark zrealizowano w następujących krokach:

- Wczytanie obu zbiorów do postaci Spark DataFrame
- Złączenie przestrzenne z wykorzystaniem zadanej metody
- Grupowanie wynikowego zbioru na podstawie kolumny z kodem TERYT przyporządkowanej gminy
- Zliczenie wystąpień dla każdej z gmin
- Przeniesienie wynikowego Spark DataFrame do postacie pandas DataFrame i zapis wyników do pliku

Praktyczna realizacja zadania realizowanego w ramach scenariusza A1 z użyciem Apache Spark jest stosunkowo nieskomplikowana. Z wykorzystaniem cyberinfrastruktury CENAGIS realizacja wszystkich kroków jest pod względem długości zbliżona do implementacji z użyciem wyłącznie biblioteki geopandas.

Uzyskanie wyników z wykorzystaniem klastra Spark o mocy odpowiadającej jednostce testowej (4 vCPU/32GB RAM) zajęło średnio 38 minut. W przypadku wykorzystania klastra o mocy 200 vCPU czas opracowania danych zajął średnio ok. 1 minutę 26 sekund. Należy zauważyć, że 50-krotny wzrost mocy klastra przełożył się w tym przypadku na ok. 26-krotne przyspieszenie obliczeń co jest związane z operacjami, którą powodują narzut na obliczenia rozproszone takimi jak inicjalizacja maszyn *worker*, transfer danych czy zapis wyników na dysku lokalnym. W tabeli 15 zaprezentowano wyniki pomiarów przeprowadzonych w tej części badań.

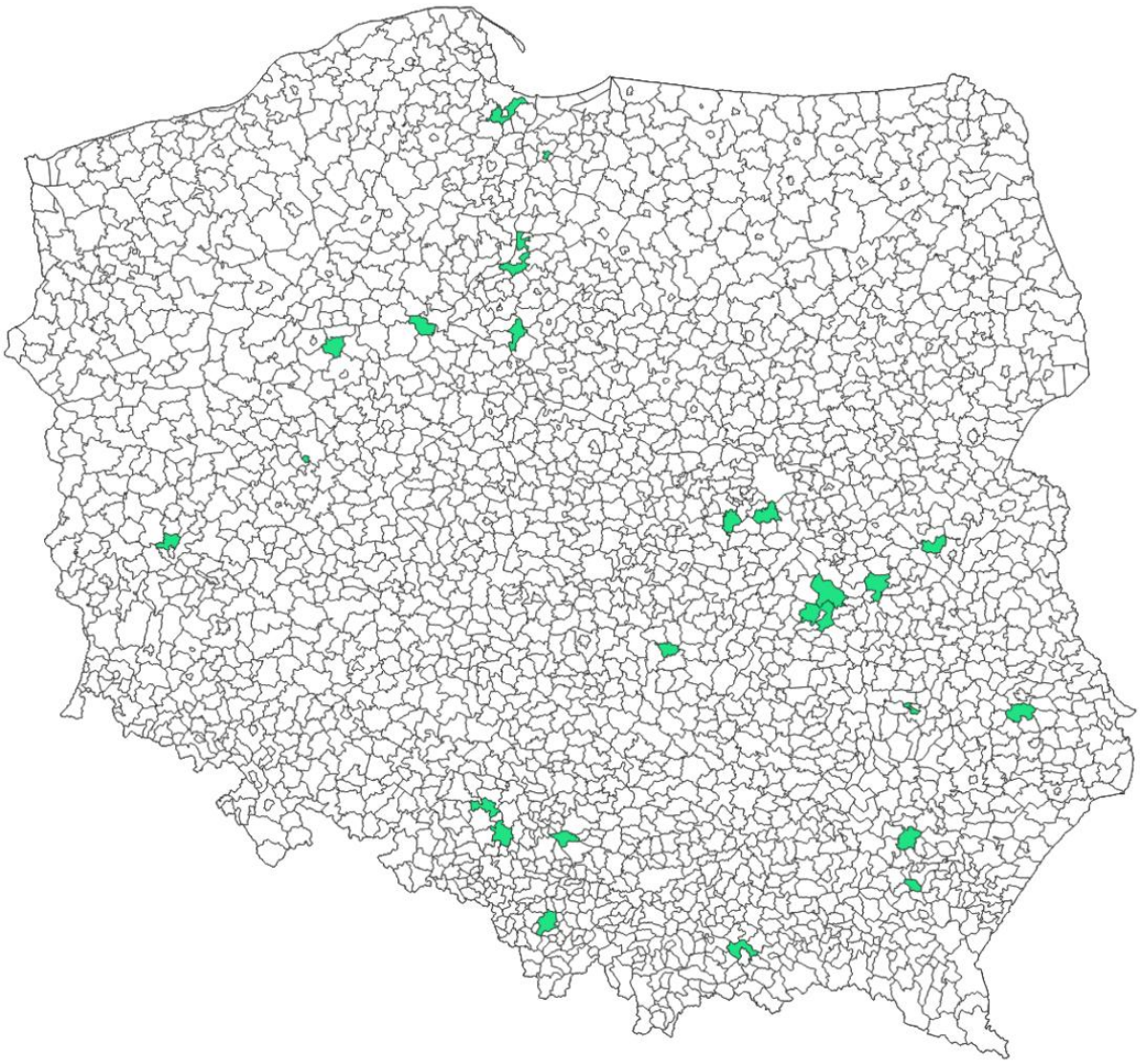
Tabela 15. Wyniki pomiarów czasu realizacji scenariusza A1 z użyciem środowiska SBD CENAGIS w dwóch wariantach klastra: 4 i 200 vCPU (opracowanie własne)

Próba	Spark (200 vCPU) [gg:mm:ss]	Spark (4 vCPU) [gg:mm:ss]
1	00:01:34	00:38:55
2	00:01:22	00:37:39
3	00:01:23	00:38:34
4	00:01:21	00:37:58
5	00:01:25	00:37:55
6	00:01:23	00:38:04
7	00:01:28	00:38:41

Próba	Spark (200 vCPU) [gg:mm:ss]	Spark (4 vCPU) [gg:mm:ss]
8	00:01:31	00:37:43
9	00:01:26	00:38:29
10	00:01:30	00:38:13
Średnia	**00:01:26**	**00:38:13**

Rezultaty ze wszystkich eksperymentów opartych o język Python, Spark oraz PostGIS zakończyły się dokładnie tym samym wynikiem tj. tą samą liczbą zliczonych punktów dla wszystkich gmin. Różnica w rezultatach pojawiła się jednak w przypadku oprogramowania QGIS oraz ArcGIS Pro. Dla oprogramowania QGIS różnica wystąpiła tylko w dwóch przypadkach: pomiędzy gminami Nowy Sącz i Chełmiec oraz wiejską i miejską gminą Malbork

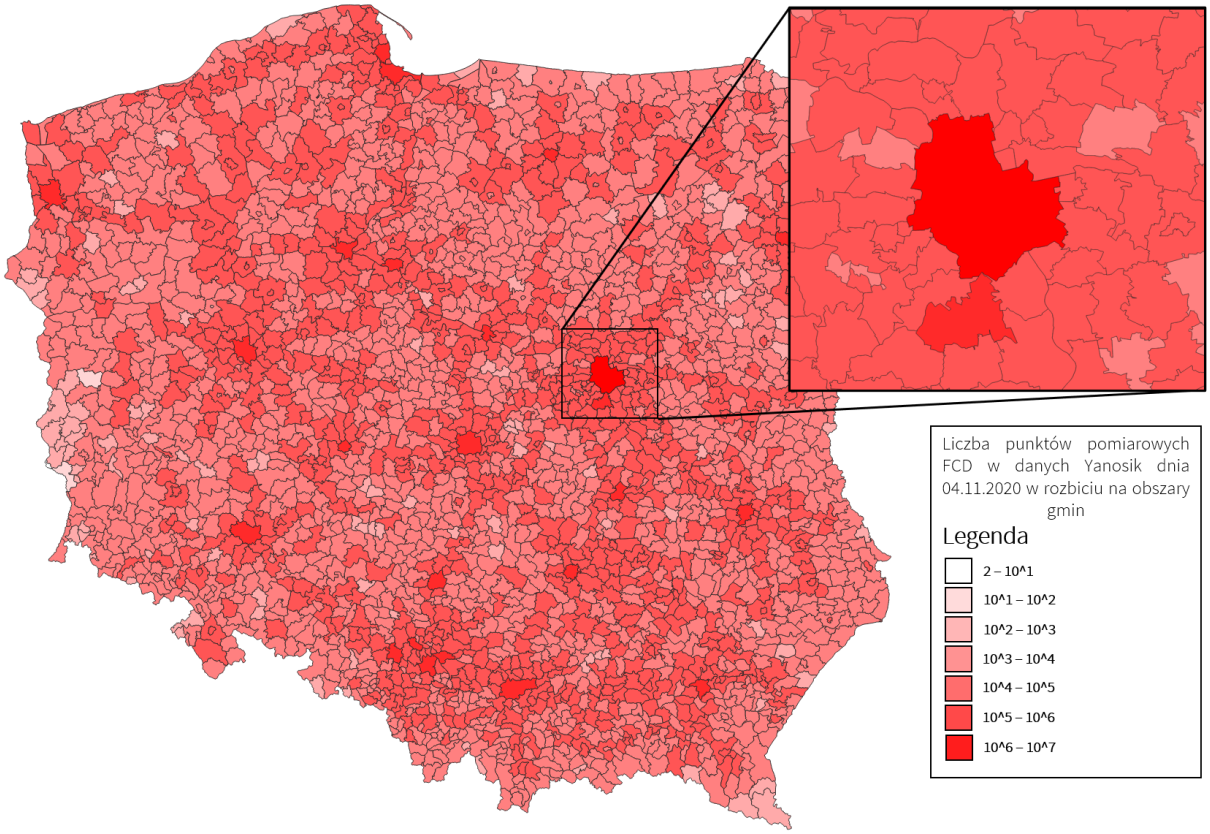
W przypadku oprogramowania ArcGIS Pro różnica jednego punktu pomiarowego wystąpiła w 24 gminach (rysunek 48), ale zawsze zliczony był o jeden punkt mniej niż z użyciem pozostałych metod. Różnica w przypadku oprogramowania QGIS wynika z przypisania jednego z punktów do innego obszaru (występuje różnica +1 i -1 na sąsiadujących obszarach). Może to być związane z błędem numerycznym bądź innym sposobem analizy topologicznej przypadków szczególnych.



Rysunek 48. Różnica w wynikach zliczenia pomiędzy opracowaniem ArcGIS Pro, a Python/Spark/PostGIS. Kolorem zielonym oznaczono spadek liczby o 1 (opracowanie własne)

Jednak, w przypadku oprogramowania ArcGIS Pro różnica nie wyrównuje się w skali całego opracowania, a więc wynik sumaryczny nie jest identyczny z tym, uzyskanym pozostałymi metodami. Oznacza to, że wynik osiągnięty w tym oprogramowaniu wykluczył część danych w trakcie przetworzeń. Zastosowany operator przestrzenny „Completely Within" teoretycznie powinien realizować to samo zadanie co operator „ST_WITHIN" bazy danych PostGIS. Możliwe jest jednak, że wewnętrzne mechanizmy oprogramowania ArcGIS stosują inne mechanizmy analizy topologii bądź inną precyzję współrzędnych. Zamknięta struktura oprogramowania utrudnia jednak dalszą analizę tych różnic.

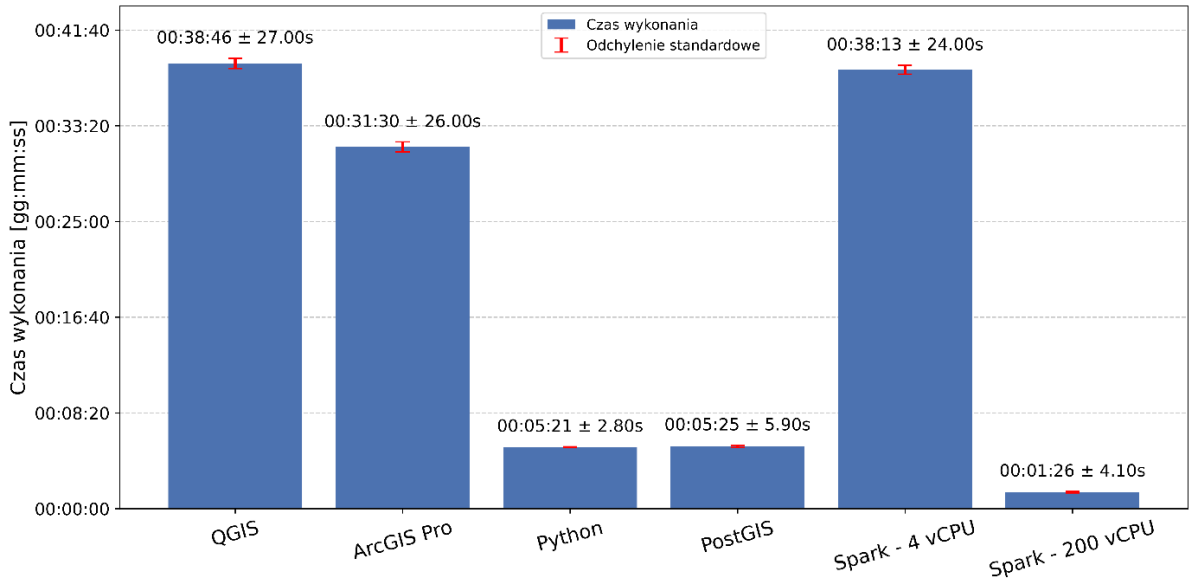
Wynik analizy przedstawiono w formie wizualizacji kartograficznej (rysunek 49). Należy zwrócić uwagę na znaczne nasilenie pomiarów w gminach miejskich oraz w gminach, przez które przebiegają główne szlaki komunikacji samochodowej.



Rysunek 49. Wynik realizacji scenariusza 1. badań porównawczych. Liczba pomiarów danych punktowych FCD z aplikacji Yanosik z dnia 04.11.2020 z rozbiciem na gminy (opracowanie własne)

Na rysunku 50 przedstawiono uśrednione czasy wykonania obliczeń ze scenariusza A1 we wszystkich badanych środowiskach. W przypadku wykorzystania pojedynczej maszyny do analizy danych, rozwiązaniami najszybszymi okazał się język Python oraz baza danych PostGIS. Czas wykonania operacji zliczenia z użyciem tych narzędzi wyniósł około 5 minut 20 sekund. Realizacja tego zadania z użyciem narzędzi desktop GIS okazała się być znacznie wolniejsza. W przypadku oprogramowania QGIS jest to blisko 40 minut, a dla ArcGIS czas ten wyniósł ponad 31 minut. Implementacja w Spark z wykorzystaniem mocy obliczeniowych analogicznych do maszyny testowej wykazała wydajność na poziomie oprogramowania desktop. Jest to związane m.in. z dodatkowym narzutem związanym z charakterystyką działania Apache Spark. Wykonanie analogicznych obliczeń z użyciem klastra o większej mocy pozwoliło na osiągnięcie średniego czasu na poziomie 86 sekund, co stanowi najszybszy zarejestrowany wynik (oczywiście należy wziąć pod uwagę fakt, że wykorzystane zostały realnie dużo większe moce

obliczeniowe). Na podstawie tych pomiarów można zasymulować czas realizacji analogicznych obliczeń dla całego zbioru (720 dni). W przypadku wykonania w języku Python analizy dzień po dniu zajęłoby to 64 godziny nie licząc czasu agregacji wyników cząstkowych. Próba wykonania takich obliczeń w oprogramowaniu QGIS zajęłaby z kolei ponad 464 godziny, czyli ponad 19 dni.



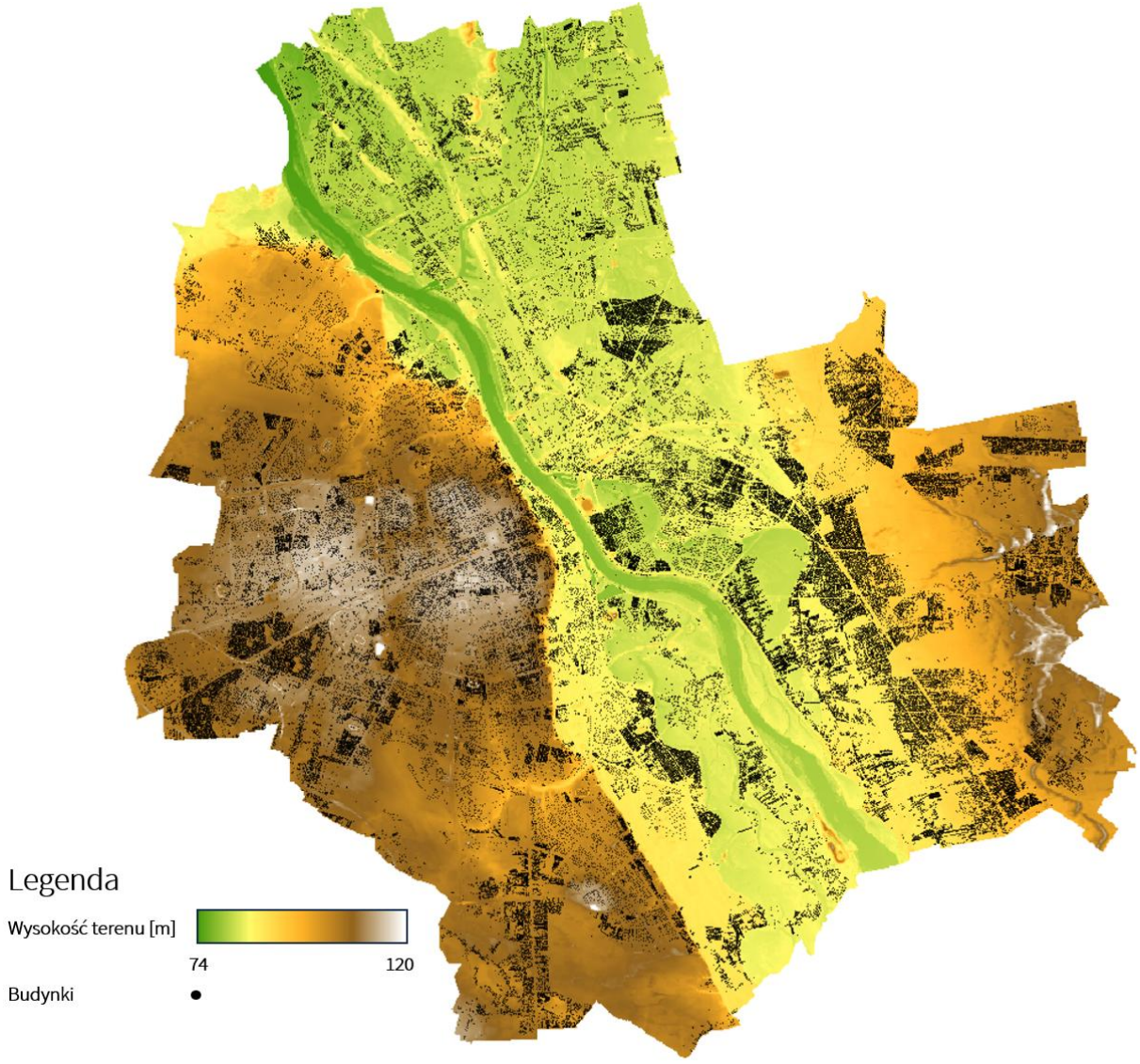
Rysunek 50. Porównanie średniego czasu wykonania scenariusza A1 w zależności od zastosowanego oprogramowania wraz z informacją o odchyleniu standardowym dla poszczególnych podejść (opracowanie własne)

9.2. Scenariusz A2 – obliczanie wysokości budynków na podstawie zNMPT

Zadanie w scenariuszu A2 polega na obliczeniu wysokości każdego z budynków na terenie Warszawy na podstawie znormalizowanego NMT (zNMT) wyliczonego z rastrów NMPT i NMT. Dane o wysokości przypisywane są jako atrybut do warstwy BUBD z Bazy Danych Obiektów Topograficznych (BDOT10k).

Scenariusz wymaga wykonania operacji wzajemnego dopasowania dwóch warstw rastrowych o różnych rozdzielczościach przestrzennych (0.5m dla NMPT oraz 1m dla NMT), złączenia przestrzennego rastra zNMPT z obrysami budynków, a także obliczenia wysokości w obrębie każdego z budynków. Metoda wyznaczania w ten sposób wysokości budynków znajduje wiele aplikacji praktycznych ((Rahman, 2014),(Liu M. i inni, 2024), (Kałuski, Hościłło i Gurdak, 2018)).

Dane rastrowe, pozyskane z zasobów GUGiK (przycięte do zakresu granic Warszawy), w postaci plików GeoTIFF w kompresji LZW[61] zajmują 3GB przestrzeni dyskowej w przypadku rastra NMPT oraz 0.8 GB dla rastra NMT co przekłada się odpowiednio na 3443.4 mln i 868.9 mln pikseli. Zbiór danych wektorowych zawiera 149 731 rekordów, co przekłada się na 2GB w postaci pliku ESRI SHAPEFILE. Na rysunku 51 przedstawiono rozkład przestrzenny analizowanych danych wektorowych na tle rastra NMPT.



Rysunek 51. Zakres analizowanych danych wektorowych oraz rastrowych – obszar Warszawy, numeryczny model pokrycia terenu oraz warstwa BUBD z bazy danych BDOT10k (opracowanie własne)

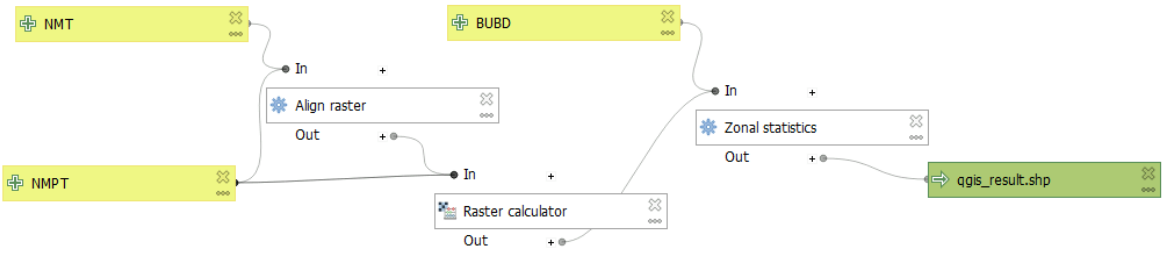
Scenariusz A2 został zrealizowany zgodnie z przyjętymi wcześniej założeniami w 4. przygotowanych środowiskach testowych: QGIS, ArcGIS Pro, Python oraz SBD CENAGIS.

[61] LZW – metoda kompresji bezstratnej Lempel-Ziv-Welch (Welch, 1984)

Realizację drugiego scenariusza badań porównawczych ponownie rozpoczęto od analizy opartej o oprogramowanie QGIS, gdzie wykorzystano narzędzie Model Designer do implementacji logiki przetworzeń (rysunek 52).

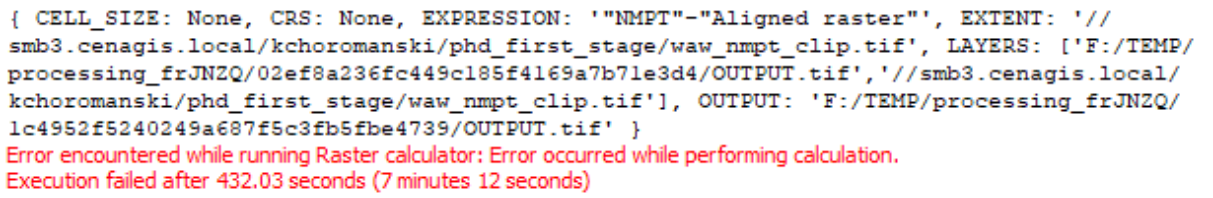
Realizacja obliczeń składała się z następujących kroków:

1. Wczytanie danych rastrowych
2. Transformacja rastra NMT do rozdzielczości i układu siatki rastra NMPT (narzędzie *Align raster*)
3. Obliczenie rastra różnicowego (narzędzie *Raster calculator*)
4. Wczytanie warstwy wektorowej BUBD
5. Obliczenie średniej wartości rastra zNMPT w obszarach budynków (narzędzie *Zonal statistics*)
6. Zapis wyniku do pliku SHP



Rysunek 52. Schemat blokowy z modułu Model Designer oprogramowania QGIS utworzony do realizacji scenariusza A2 (opracowanie własne)

W procesie realizacji scenariusza w oprogramowaniu QGIS wystąpiły problemy związane głównie z niską wartością informacyjną komunikatów o występujących błędach. Większość problemów związana była z wysycaniem przestrzeni dyskowej na etapie zapisu danych tymczasowych. QGIS zapisuje wyniki każdego etapu w postaci pliku tymczasowego na dysku, bez możliwości bezpośredniego sprecyzowania ich parametrów (np. kompresji). W wyniku tego, do przeprowadzenia opisywanego powyżej scenariusza konieczne jest zarezerwowanie przynajmniej 25.7 GB wolnego miejsca na dysku. Komunikat informujący o niepowodzeniu nie zawierał jednak informacji o właściwym źródle problemu (rysunek 53). Dodatkowo zapis wszystkich danych tymczasowych na dysku wydłuża proces obliczeń.



Rysunek 53. Przykładowy komunikat błędu zwracany przez oprogramowanie QGIS spowodowany wysyceniem się miejsca na dysku w związku z zapisywaniem danych tymczasowych. Komunikat nie podaje właściwego źródła problemu (opracowanie własne)

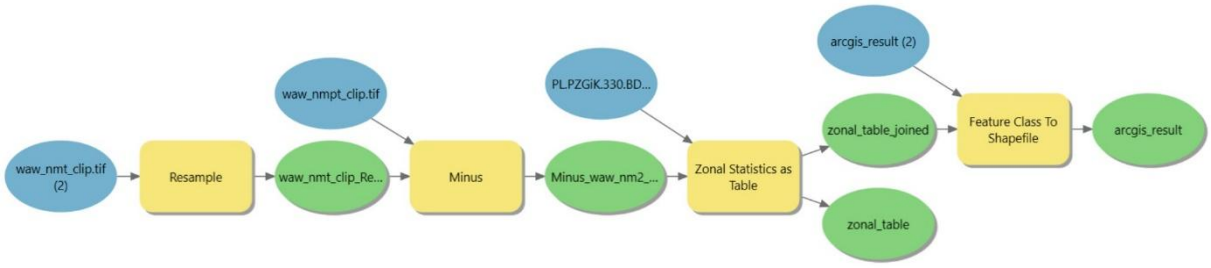
Realizacja przetworzeń danych rastrowych w oprogramowaniu QGIS zajęła średnio 31 minut 41 sekund. Najdłuższym procesem było obliczenie wartości średniej w poligonach warstwy BUBD na podstawie rastra zNMPT. Jest to zrozumiałe, gdyż w ramach tego procesu realizowane było przestrzenne złączenie danych rastrowych z poligonowymi. Czas obliczeń w poszczególnych etapach zaprezentowano w tabeli 16.

Tabela 16. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A2 z wykorzystaniem oprogramowania QGIS z rozbiciem na poszczególne etapy przetwarzania (opracowanie własne)

Próba	Dostosowanie rastrów [gg:mm:ss]	Tworzenie rastra z NMPT [gg:mm:ss]	Obliczenie wartości średniej wysokości [gg:mm:ss]	Całość [gg:mm:ss]
1	00:07:02	00:04:26	00:21:09	00:32:37
2	00:06:58	00:04:34	00:23:13	00:34:45
3	00:07:16	00:04:31	00:23:32	00:35:19
4	00:06:50	00:03:32	00:14:47	00:25:09
5	00:06:58	00:03:50	00:20:06	00:30:54
6	00:06:55	00:03:45	00:19:02	00:29:42
7	00:06:59	00:03:59	00:20:21	00:31:19
8	00:06:56	00:04:17	00:21:33	00:32:46
9	00:07:05	00:04:03	00:21:58	00:33:06
10	00:07:18	00:04:14	00:19:38	00:31:10
Średnia	**00:07:02**	**00:04:07**	**00:20:32**	**00:31:41**

Podobnie, jak w przypadku scenariusza A1, w oprogramowaniu ArcGIS Pro użyto narzędzia *ModelBuilder* w celu przygotowania algorytmu obliczeń dla scenariusza A2 (rysunek 54). Realizacja tego scenariusza sprowadziła się do wykonania następujących kroków:

1. Odczyt pliku SHP z warstwą BUBD
2. Odczyt NMT oraz NMPT w postaci plików GeoTIFF
3. Transformacja i zwiększenie rozdzielczości rastra NMT do układu siatki rastra NMPT (narzędzie *Resample*)
4. Utworzenie rastra różnicowego (narzędzie *Minus*)
5. Obliczenie wartości średniej rastra różnicowego w obrębie budynków (narzędzie *Zonal Statistics as Table* z ustawieniem liczenia statystyki „*Mean*")
6. Zapis wyniku do postaci pliku SHP



Rysunek 54. Schemat blokowy z modułu ModelBuilder oprogramowania ArcGIS Pro utworzony do realizacji scenariusza A2 (opracowanie własne)

Utworzenie prawidłowego wyniku wymagało wprowadzenia dodatkowych ustawień środowiskowych do oprogramowania ArcGIS Pro. Było to związane ze sposobem działania narzędzia „*Resample*". Ustawiono parametr „*Snap Raster*" oraz „*Extent*" na raster NMPT, aby wynik transformacji rastra NMT był przestrzennie dopasowany do rastra NMPT w celu uniknięcia błędów na etapie wytwarzania rastrów różnicowych. Czas obliczeń w tym wypadku był dłuższy niż w oprogramowaniu QGIS i wyniósł ok. 34 minuty. Poniżej zaprezentowano czasy pomiarów w poszczególnych próbach.

Tabela 17. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A2 z wykorzystaniem oprogramowania ArcGIS Pro z rozbiciem na poszczególne etapy przetwarzania (opracowanie własne)

Próba	Dostosowanie rastrów [gg:mm:ss]	Tworzenie rastra zNMPT [gg:mm:ss]	Obliczenie wartości średniej [gg:mm:ss]	Zapis do SHP [gg:mm:ss]	SUMA [gg:mm:ss]
1	00:11:49	00:13:13	00:03:25	00:05:31	00:33:57
2	00:11:59	00:13:21	00:03:46	00:05:08	00:34:14
3	00:11:15	00:13:21	00:04:13	00:06:19	00:35:08
4	00:12:02	00:12:22	00:03:01	00:06:01	00:33:26
5	00:11:15	00:13:14	00:03:18	00:05:17	00:33:04
6	00:12:20	00:13:31	00:04:53	00:04:11	00:34:55
7	00:11:27	00:13:04	00:04:00	00:04:53	00:33:24
8	00:12:00	00:12:32	00:04:24	00:04:52	00:33:48
9	00:11:58	00:13:54	00:04:13	00:05:01	00:35:05
10	00:11:49	00:13:14	00:03:43	00:05:00	00:33:46
Średnia	00:11:49	00:13:13	00:03:25	00:05:31	00:33:57

Analiza w środowisku Python została zrealizowana poprzez wykonanie następujących kroków:

1. Odczyt pliku SHAPEFILE z budynkami (biblioteka geopandas)
2. Odczyt NMT oraz NMPT (biblioteki rioxarray[62] i rasterio[63])
3. Transformacja i zwiększenie rozdzielczości rastra NMT do układu siatki rastra NMPT (złączenie przestrzenne rastrów – biblioteka rioxarray)
4. Utworzenie rastra różnicowego (biblioteka NumPy[64])
5. Obliczenie wartości średniej z rastra różnicowego w obrysach budynków (biblioteka rasterstats[65])
6. Zapis wyników do pliku SHAPEFILE (biblioteka geopandas)

[62] https://corteva.github.io/rioxarray/stable (data dostępu: 15.12.2025)
[63] https://rasterio.readthedocs.io/en/stable (data dostępu: 15.12.2025)
[64] https://numpy.org (data dostępu: 15.12.2025)
[65] https://pythonhosted.org/rasterstats/ (data dostępu: 15.12.2025)

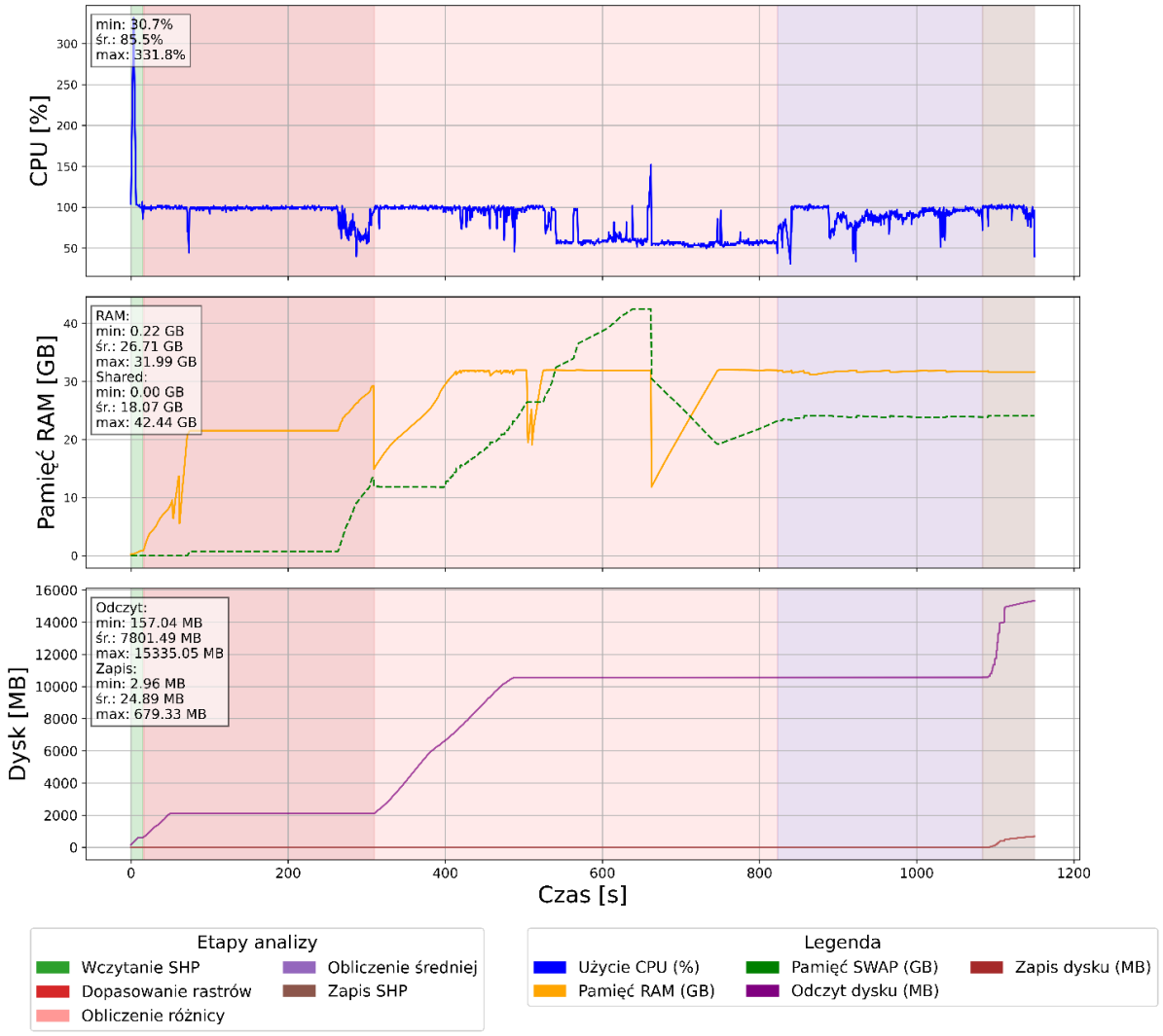
Cały kod opiera się o istniejące biblioteki, nie wymaga ręcznej implementacji rozwiązań programistycznych i w podstawowej wersji zajmuje ok. 20 linijek. Średni czas obliczeń w scenariuszu A2 z użyciem języka Python wyniósł około 18 minut 30 sekund (szczegółowy czas obliczeń poszczególnych etapów przedstawiono w tabeli 18). Najdłuższymi procesami było obliczenie rastrów różnicowych, dopasowanie rastrów oraz obliczenie średniej w obszarach budynków. Należy jednak zauważyć na wyraźnie krótki czas wykonania operacji wczytania danych rastrowych. Jest to związane z faktem iż wykorzystywana biblioteka rioxarray wczytuje dane w sposób „leniwy", a więc dopiero w momencie kiedy są one niezbędnie potrzebne do obliczeń. W związku z tym wydłużone czasy dopasowania rastrów i obliczenia rastra różnicowego są związane z faktem, że w tych procesach następuje realne załadowanie danych do pamięci RAM.

W porównaniu z oprogramowaniem QGIS zauważyć można znacznie krótszy czas potrzebny na obliczenie wartości średniej w poligonach. Natomiast czas operacji rastrowych zajął z użyciem Python więcej czasu (średnio 735 sekund w porównaniu do 671 sekund w QGIS).

Tabela 18. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A2 zaimplementowanego w języku programowania Python z rozbiciem na poszczególne etapy przetwarzania (opracowanie własne)

Próba	Wczytanie SHP [gg:mm:ss]	Wczytanie rastrów [gg:mm:ss]	Dopasowanie rastrów [gg:mm:ss]	Obliczenie różnicy [gg:mm:ss]	Obliczenie średniej [gg:mm:ss]	Aktualizacja atrybutów [gg:mm:ss]	Zapis SHP [gg:mm:ss]	Suma [gg:mm:ss]
1	00:00:18	00:00:00	00:04:54	00:08:33	00:04:21	00:00:00	00:01:07	00:19:12
2	00:00:17	00:00:01	00:03:18	00:06:40	00:04:08	00:00:00	00:00:58	00:15:23
3	00:00:16	00:00:00	00:03:24	00:09:39	00:04:27	00:00:00	00:00:59	00:18:47
4	00:00:15	00:00:00	00:03:16	00:06:40	00:04:56	00:00:00	00:01:02	00:16:09
5	00:00:17	00:00:00	00:03:28	00:09:27	00:04:25	00:00:00	00:00:57	00:18:34
6	00:00:14	00:00:00	00:03:19	00:08:60	00:05:51	00:00:00	00:01:13	00:19:36
7	00:00:17	00:00:01	00:03:28	00:09:36	00:06:28	00:00:00	00:00:59	00:20:49
8	00:00:15	00:00:00	00:03:20	00:08:42	00:05:44	00:00:00	00:00:59	00:19:01
9	00:00:16	00:00:00	00:03:30	00:10:07	00:04:28	00:00:00	00:00:58	00:19:21
10	00:00:15	00:00:00	00:03:19	00:08:52	00:04:31	00:00:00	00:01:02	00:18:01
Średnia	**00:00:16**	**00:00:00**	**00:03:32**	**00:08:44**	**00:04:56**	**00:00:00**	**00:01:01**	**00:18:29**

Analiza zużycia zasobów (rysunek 55) jasno uwidacznia fakt, iż pełen odczyt danych rastrowych zaszedł dopiero na etapie obliczenia rastra różnicowego, co potwierdza leniwą ewaluacje obliczeń przez rioxarray. Zauważyć należy także wykorzystanie pamięci *swap* spowodowane pełnym wysyceniem dostępnej pamięci RAM. Zastosowanie w tym przypadku większej ilości pamięci spowodowałoby przyspieszenie obliczeń z użyciem proponowanego podejścia. Zdecydowana większość obliczeń (z wyłączeniem procesu wczytywania SHP) przeprowadzana jest z użyciem pojedynczego wątku procesora.



Rysunek 55. Wykres wykorzystania poszczególnych zasobów (CPU, RAM, Dysk) w czasie trwania obliczeń dla scenariusza A2 w języku programowania Python (opracowanie własne)

Implementacja analizy na danych rastrowych z użyciem Apache Spark była dużym wyzwaniem i wymagała wprowadzenia wielu dedykowanych rozwiązań. Przed wykonaniem właściwych obliczeń konieczne było przeniesienie danych wejściowych na przestrzeń HDFS. Rastry

w formacie GeoTIFF (NMT i NMPT) przekopiowano bezpośrednio. W przypadku warstwy BUBD dokonano wcześniej transformacji danych do postaci GeoMesa-FS, co wiązało się z koniecznością zmiany układu współrzędnych na układ WGS84. Ostatecznie analiza sprowadziła się do realizacji następujących kroków:

1. Odczyt zbioru danych BUBD w postaci GeoMesa-FS
2. Odczyt rastrów NMPT i NMT do postaci Spark DataFrame ze wsparciem RasterFrames
3. Transformacja geometrii zbioru danych BUBD z układu WGS 84 do układu PL-1992
4. Transformacja rastra NMT do układu rastra NMPT z użyciem biblioteki GeoTrellis i nadanie identyfikatora przestrzennego każdemu z kafelków
5. Konwersja rastra NMPT do postaci GeoTrellis i nadanie identyfikatora przestrzennego każdemu z kafelków
6. Złączenie rastrów NMT i NMPT na podstawie identyfikatorów przestrzennych nadanych z użyciem biblioteki GeoTrellis
7. Konwersja rastrów do DataFrame w postaci RasterFrames
8. Obliczenie wartości zNMPT poprzez odjęcie wartości rastrów NMPT i NMT w poszczególnych kafelkach
9. Ekstrakcja obwiedni kafelków rastra zNMPT do warstwy wektorowej
10. Złączenie lewostronne zbioru BUBD z obwiedniami kafelków rastra zNMPT (operator ST_INTERSECTS)
11. Rasteryzacja danych wektorowych BUBD w przestrzeni odpowiadających im kafelków
12. Maskowanie wartości rastra zNMPT na rastrową postać danych BUBD
13. Obliczenie wartości średniej z pikseli w rastrowej postaci danych BUBD oraz wyliczenie liczby pikseli reprezentowanych przez dany budynek w danym kafelku rastra
14. Obliczenie ostatecznej wartości średniej dla każdego z budynków poprzez średnią ważoną wartości średniej z każdego kafelka oraz liczby pikseli reprezentujących danych budynek w tym kafelku

W przypadku tego scenariusza konieczne było podjęcie szeregu dodatkowych czynności, które nie były wymagane w przypadku innych narzędzi GIS. Odpowiednie złączenie przestrzenne rastrów NMPT i NMT wymagało wykorzystania dwóch bibliotek SBD (RasterFrames i GeoTrellis). W odróżnieniu od innych metod nie istniała możliwość bezpośredniego zliczenia wartości rastra w zakresie danych wektorowych. Konieczna była wcześniejsza rasteryzacja tych danych w przestrzeni kafelków RasterFrames. Dodatkowo należało uwzględnić przypadek, gdzie budynek znajduje się na granicy kafelków i wartość średnia nie zostanie obliczona dla całego

jego obszaru. Dodatkowym czynnikiem wydłużającym czas był odczyt metadanych rastra, który wymagał jego całkowitego odczytu z HDFS. Pomimo, iż biblioteka RasterFrames posiada funkcję łączenia rastrów, to nie obsługuje ona metod interpolacji innych niż najbliższy sąsiad (w wersji wspieranej w środowisku CENAGIS), a także jej wewnętrzne mechanizmy obliczeniowe utrudniają wykorzystanie tej funkcji w przypadku łączenia większych zbiorów danych (wymuszane jest zastosowanie mechanizmu *broadcast*[66]). W związku z tym konieczne było wykorzystanie mechanizmów biblioteki GeoTrellis, co wiązało się z koniecznością transformacji danych pomiędzy formatami obu bibliotek.

Czas analizy na danych rastrowych z użyciem Apache Spark znacząco odbiegł o innych wykorzystywanych metod. Nawet przy zastosowaniu 200 vCPU czas ten był wyższy niż wszystkie inne podejścia. Jest to związane z koniecznością wykonania dodatkowych obliczeń takich jak konwersja pomiędzy typami danych, transformacje układów czy odczyt metadanych. Z przedstawionego czasu, średnio 545 sekund zajął odczyt metadanych z poszczególnych kafelków. Wymagał on analizy całego zbioru danych w domyślnej konfiguracji, ale ten proces mógłby zostać pominięty, gdyby możliwe było odczytanie metadanych zbioru rastrowego tak, jak jest to realizowane z wykorzystaniem innych środowisk. Zastosowano w tym przypadku tryb *cache*[67], który przyspieszył dalsze obliczenia, lecz nadal ich czas znacząco odbiega od innych rozwiązań. Tak wysoki czas, nawet przy dużej mocy klastra wskazuje na bardzo duże pole do optymalizacji poszczególnych operacji. W tabeli 19 przedstawiono szczegółowe czasu wykonania scenariusza z użyciem środowiska SBD CENAGIS.

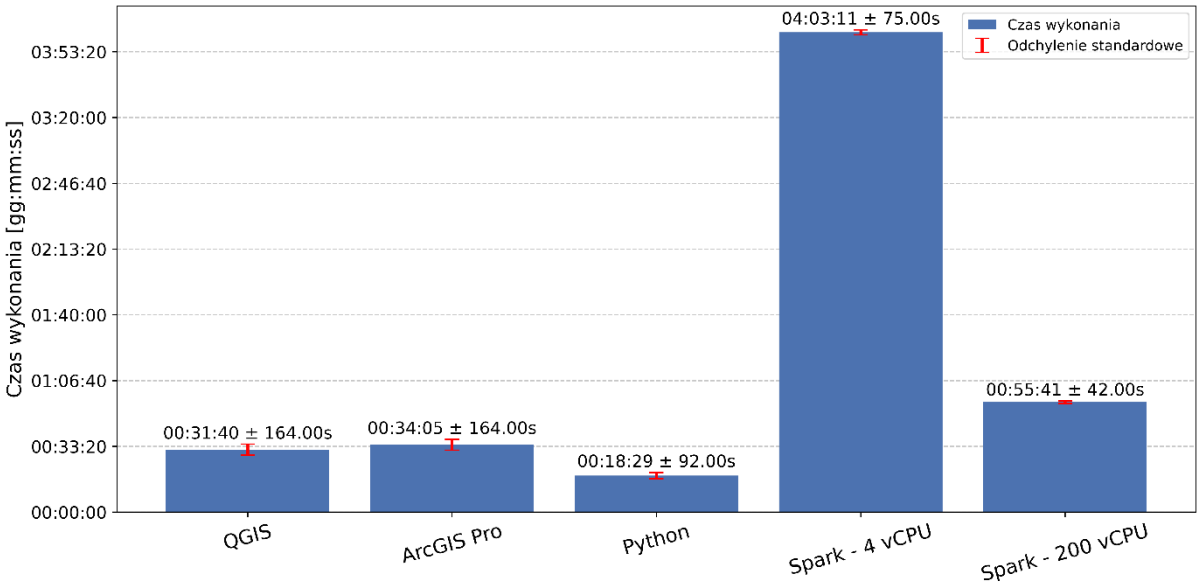
[66] Mechanizm *broadcast* w Apache Spark polega na jednorazowym rozesłaniu niewielkiego zbioru danych do pamięci operacyjnej wszystkich maszyn *worker*, co umożliwia lokalne wykorzystanie tych danych w zadaniach obliczeniowych bez konieczności kosztownej komunikacji sieciowej w trakcie wykonywania operacji.

[67] Tryb umożliwiający przechowywanie w pamięci operacyjnej maszyn *worker* danych odczytanych wcześniej z systemu plików. Dzięki temu zbiór wczytany na potrzeby analizy metadanych kafli nie musiał być ponownie odczytywany z HDFS podczas właściwych obliczeń, lecz był pobierany bezpośrednio z pamięci operacyjnej maszyn.

Tabela 19. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A2 zaimplementowanego w środowisku Spark z rozbiciem na wykorzystaną moc klastra obliczeniowego (opracowanie własne)

Próba	pySpark (200 vCPU) [gg:mm:ss]	pySpark (4 vCPU) [gg:mm:ss]
1	00:56:25	04:04:01
2	00:54:51	04:04:35
3	00:55:44	04:03:15
4	00:54:53	04:04:36
5	00:55:19	04:00:46
6	00:56:05	04:01:32
7	00:55:20	04:04:14
8	00:54:58	04:03:32
9	00:56:17	04:03:09
10	00:57:00	04:02:07
Średnia	**00:55:41**	**04:03:11**

Jak widać na zestawieniu podsumowującym pomiary w ramach scenariusza A2 (rysunek 56) i w tym przypadku rozwiązanie oparte o język Python okazało się być najwydajniejsze. Oba testowane pakiety oprogramowania desktop GIS osiągnęły zbliżone wyniki, blisko 2 razy wolniejsze względem implementacji w Python. Zdecydowanie najgorsze pod względem wydajności wyniki zaprezentowało środowisko SBD, gdzie nawet przy zastosowaniu znacznie większych mocy obliczeniowych czas obliczeń był znacznie dłuższy.



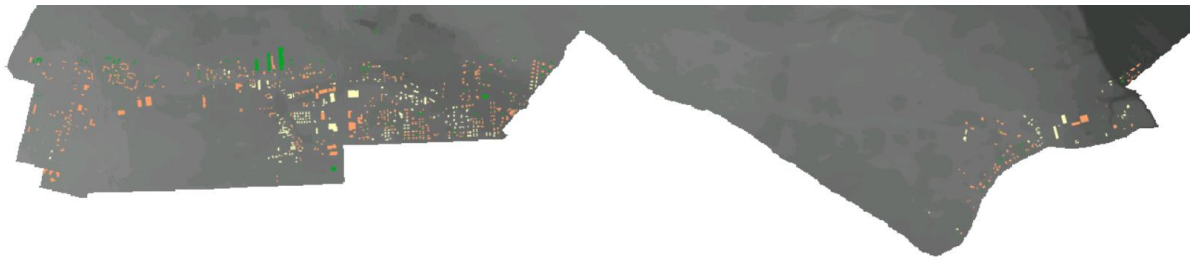
Rysunek 56. Porównanie średniego czasu wykonania scenariusza A2 w zależności od zastosowanego oprogramowania wraz z informacją o odchyleniu standardowym dla poszczególnych podejść (opracowanie własne)

Po przeprowadzeniu badań wydajnościowych z użyciem wszystkich czterech pakietów oprogramowania wykonano analizę spójności uzyskanych wyników. Łącznie, informacja o średniej wysokości została przypisana do 155 307 budynków. Sprawdzono różnice pomiędzy przypisanymi wartościami dla poszczególnych zbiorów (tabela 20). Za zbiór referencyjny przyjęto wynik uzyskany z użyciem języka Python, a za wynik różniący się przyjęto wynik odbiegający od średniej o więcej niż 1cm. Niemal wszystkie obiekty w przypadku narzędzi QGIS, ArcGIS i Python charakteryzowały się identycznymi wartościami obliczonej wysokości. W przypadku oprogramowania QGIS rozbieżne wartości wystąpiły dla 92 obiektów, natomiast w ArcGIS dla 126, co stanowi mniej niż 0,1% całego zbioru. Dużo bardziej zauważalna różnica została zidentyfikowana w przypadku wyniku uzyskano z użyciem Spark. Blisko 7000 obiektów cechowało się inną wartością.

Tabela 20. Podsumowanie różnic pomiędzy wynikami scenariusza A2 z rozbiciem na poszczególne pakiety oprogramowania względem wyników uzyskanych w oprogramowaniu Python (opracowanie własne)

	Python	QGIS	ArcGIS Pro	Spark
Liczba obiektów	155 307	155 307	155 307	155 306
Procent zgodności	-	99,94%	99,91%	95,52%
Liczba obiektów o innej wartości	-	92	126	6955
Największa różnica (m)	-	7,11	0,3	8,39
Mediana różnic (m)	-	0,021	0,017	0,015
Średnia różnica (m)	-	0,239	0,029	0,121

Wszystkie znaczne (powyżej 0.1m) różnice w przypadku wyniku uzyskanego z użyciem Spark skoncentrowały się w dolnej części obszaru zainteresowania (rysunek 57). Równy przebieg linii występowania błędów sugeruje, że kafelki rastrów u dołu opracowania nie złączyły się poprawnie. Może być to związane z problemem z wewnętrznymi mechanizmami bibliotek RasterFrames bądź GeoTrellis – są to elementy najniższego rzędu kafelków w opracowaniu. W tym rejonie znajduje się też jedyny budynek, do którego nie została przypisana żadna wartość w tym wariancie obliczeń, a który uzyskał poprawne wartości z użyciem innych narzędzi. Błędów grubych u dołu obszaru opracowania jest łącznie 870. Pozostałe błędy (poniżej 0.1m) występują w równomiernym rozproszeniu na całym obszarze opracowania. Wynikają więc prawdopodobnie z różnic w systemie transformacji i łączenia rastrów, a nie uśredniania wartości pomiędzy kafelkami.

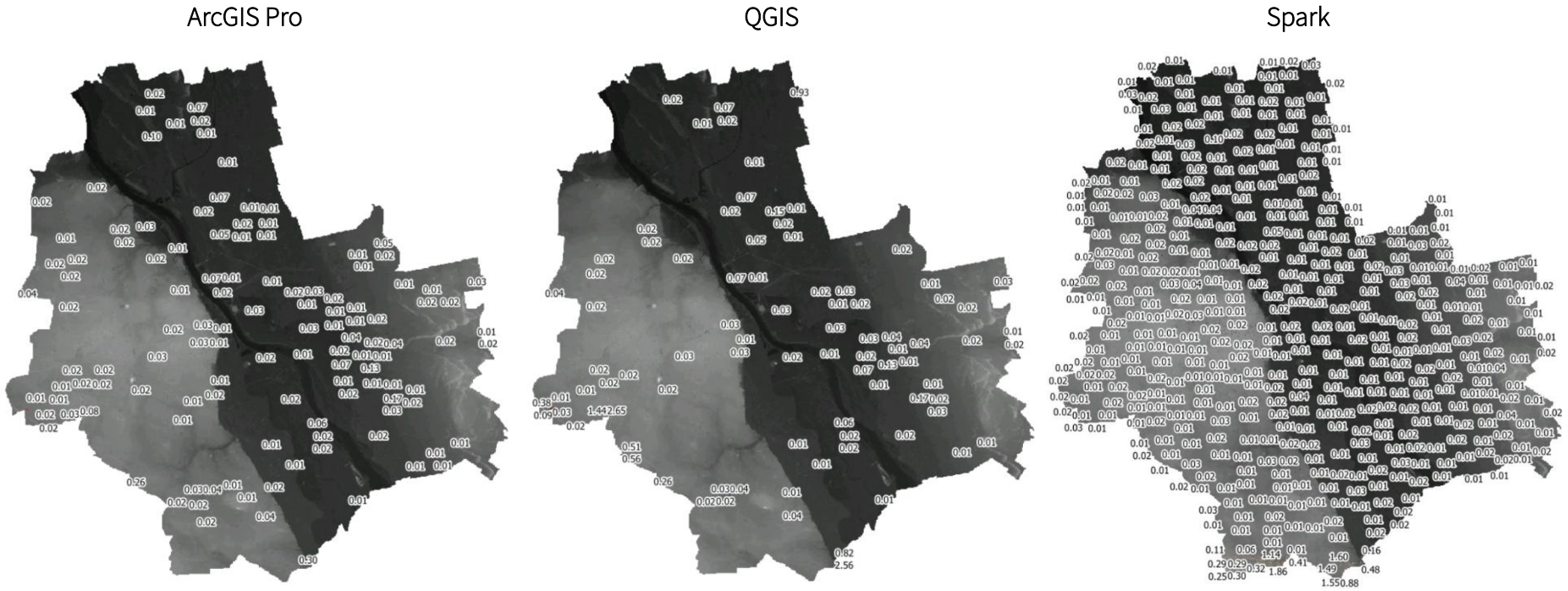


Rysunek 57. Wizualizacja różnic pomiarowych w dolnej części opracowania. Kolorem zielonym oznaczono błąd poniżej 0.1m, żółtym od 0.1 do 1m, a pomarańczowym powyżej 1m (opracowanie własne)

W przypadku pozostałych trzech metod, największe różnice znajdowały się na granicy warstw wysokościowych. W przykładzie różnic pomiędzy warstwą Python, a QGIS, widać, że rekordowa różnica (7.11m) padła dla obiektu, którego większość znajduje się poza zakresem występowania warstw wysokościowych (rysunek 58). Sposób interpolacji warstw na granicach oraz wartości przypisywane brakującym pikselom przez poszczególne narzędzia muszą się zatem między sobą różnić.

Rysunek 58. Wizualizacja błędów pomiarowych w oprogramowaniu QGIS na granicach rastra. Do każdego obiektu przypisano różnicę (w metrach) pomiędzy wynikami uzyskanym z użyciem języka programowania Python, a oprogramowania QGIS (opracowanie własne)

Niewielkie różnice (poniżej 10 cm) pojawiły się w przypadku wszystkich badanych pakietów oprogramowania. Na rysunku 59 zwizualizowane lokalizację i wielkość różnic w zakresie całego opracowania w porównaniu do wyników uzyskanych w implementacji w języku Python. W przypadku wyników pochodzących z oprogramowania desktop GIS oraz języka Python różnice występują w niewielkiej liczbie przypadków, zwykle w zbliżonych lokalizacjach. W przypadku implementacji w Spark różnice rozłożone są równomiernie na całym obszarze opracowania.



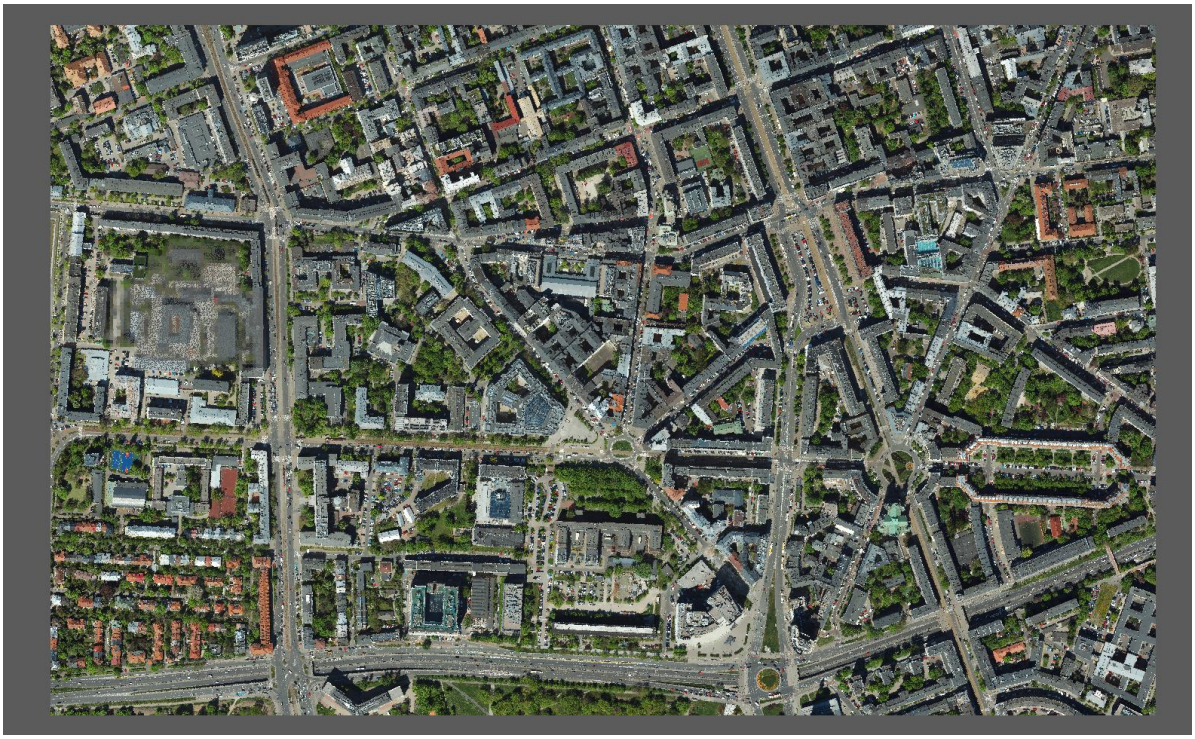
Rysunek 59. Zestawienie różnic wartości uzyskanych dla poszczególnych obiektów wektorowych w ramach realizacji scenariusza A2 w odniesieniu do wyników realizacji tego scenariusza w języku programowania Python (opracowanie własne)

9.3. Scenariusz A3 – obliczanie wysokości budynków na podstawie chmury punktów

Zadaniem realizowanym w scenariuszu A3 było obliczenie wysokości budynków na terenie okolic Kampusu Głównego Politechniki Warszawskiej na podstawie chmury punktów z zasobów GUGiK (ok. 1 GB w formie plików LAZ, 4 arkusze, ok. 213 mln punktów). Informacje o wysokości przypisano jako atrybut do warstwy BUBD z bazy danych BDOT10k (ok. 2 GB pliku SHAPEFILE, 149 731 rekordów).

Obliczenie wysokości budynków z wykorzystaniem chmur punktów pozyskanych z pułapu lotniczego stanowi jeden z popularnych zastosowań tych danych ((Cao i inni, 2020), (Rahman, 2014)). Scenariusz zakłada przeprowadzenie filtracji atrybutowej chmury punktów do klasy 6, odpowiadającej klasie *Building* w klasyfikacji ISPRS zgodnej ze standardem LAS (Open Geospatial Consortium, 2008), następnie wykonanie złączenia przestrzennego chmury punktów z wektorową warstwą budynków oraz obliczenie różnicy wysokości punktów zlokalizowanych wewnątrz poligonów reprezentujących budynki. Na rysunku 60 pokazano zasięg przestrzenny danych 3D poddanych analizie.

W przypadku praktycznego zastosowania chmury punktów do obliczenia wysokości budynków stosuje się najczęściej operację rasteryzacji chmury. W analizowanym scenariuszu wykorzystano bardziej podatną na błędy metodę analizy lokalizacji punktów (odjęcie najniższego od najwyższego punktu dla każdego budynku) w celu przeprowadzenia całości scenariusza wyłącznie z użyciem chmur punktów, a nie rastrów).



Rysunek 60. Zakres analizowanych danych 3D w formie plików LAZ. Chmura punktów LiDAR z zasobów GUGiK dla terenu okolic Kampusu Głównego Politechniki Warszawskiej (opracowanie własne)

Scenariusz A3 został zaimplementowany w 4. środowiskach: QGIS, ArcGIS Pro, Python oraz SBD CENAGIS.

W oprogramowaniu QGIS w wersji 3.34.5 nie istnieją narzędzia pozwalające na przeprowadzenie założonej analizy w całości w oparciu o chmurę punktów. W celu uzyskania oczekiwanych wyników zastosowano więc podejście łączone, gdzie chmura punktów w formacie LAS 1.2[68] została odczytana, przefiltrowana oraz wyeksportowana do formatu GeoPackage[69], a dalsze czynności zostały przeprowadzone jak dla klasycznych danych wektorowych.

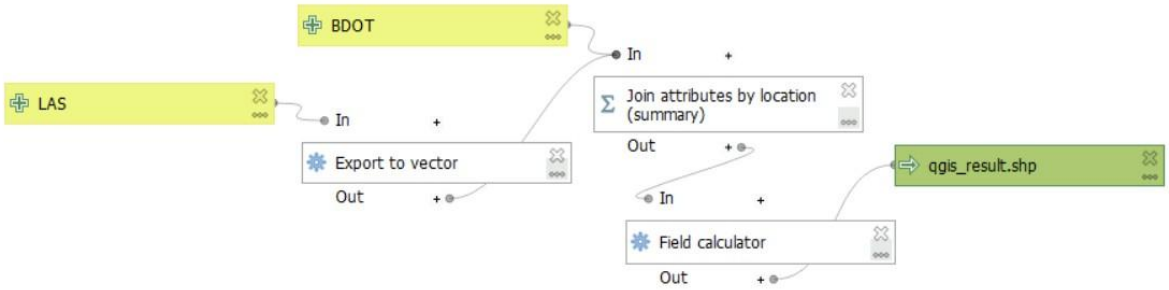
Proces analizy w oprogramowaniu QGIS sprowadził się więc do następujących kroków:

1. Wczytanie chmury punktów w formacie LAS
2. Eksport danych do formatu GeoPackage z filtracją klasy 6 (narzędzie *Export to vector*)
3. Wczytanie warstwy BUBD w formacie SHAPEFILE

[68] Binarny format przechowywania chmur punktów utrzymywany i rozwijany przez Amerykańskie Towarzystwo Fotogrametrii i Teledetekcji – ISPRS (Open Geospatial Consortium, 2008)
[69] Otwarty format przechowywania danych przestrzennych utrzymywany i rozwijany przez Open Geospatial Consortium: https://www.geopackage.org/spec140/index.html

4. Obliczenie wartości minimalnej i maksymalnej współrzędnej Z w obrębie każdego z poligonów z warstwy BUBD (narzędzie *Join attributes by location (summary)*)
5. Obliczenie różnicy wartości maksymalnej i minimalnej (narzędzie *Field Calculator*)
6. Zapis wynikowego pliku SHAPEFILE do pliku

Standardowo, proces wdrożono z użyciem narzędzia Model Designer (rysunek 61).



Rysunek 61. Schemat blokowy z modułu *Model Designer* oprogramowania QGIS utworzony do realizacji scenariusza A3 (opracowanie własne)

Czas obliczeń w oprogramowaniu QGIS, pomimo braku bezpośredniego wsparcia dla chmur punktów okazał się być dość krótki. Średnio, przeprowadzenie całej analizy zajęło 10 minut 53 sekundy. Operacją, która zajęła najwięcej czasu było zliczenie statystyk w poligonach. Szczegółowy czas trwania obliczeń w poszczególnych próbach przedstawiono w tabeli 21.

Tabela 21. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A3 przeprowadzonego z użyciem oprogramowania QGIS z rozbiciem na poszczególne etapy przetwarzania.

Próba	Filtracja i eksport do wektora [gg:mm:ss]	Obliczenie statystyk w poligonach [gg:mm:ss]	Obliczenie różnicy i zapis wyniku [gg:mm:ss]	Całość [gg:mm:ss]
1	00:04:15	00:06:17	00:00:21	00:10:54
2	00:04:10	00:06:17	00:00:21	00:10:48
3	00:04:13	00:06:14	00:00:24	00:10:50
4	00:04:22	00:06:15	00:00:23	00:11:00
5	00:04:18	00:06:20	00:00:21	00:10:59
6	00:04:16	00:06:14	00:00:22	00:10:51
7	00:04:12	00:06:19	00:00:23	00:10:53
8	00:04:11	00:06:16	00:00:22	00:10:49
9	00:04:13	00:06:19	00:00:22	00:10:54
10	00:04:11	00:06:14	00:00:24	00:10:50
Średnia	**00:04:14**	**00:06:17**	**00:00:22**	**00:10:53**

Brak dedykowanych narzędzi do realizacji tego typu obliczeń na chmurach punktów obniża intuicyjność wykonania analizy z użyciem oprogramowania QGIS. Jednak po konwersji danych do formatu wektorowego wdrożenie dalszych kroków jest nieskomplikowane.

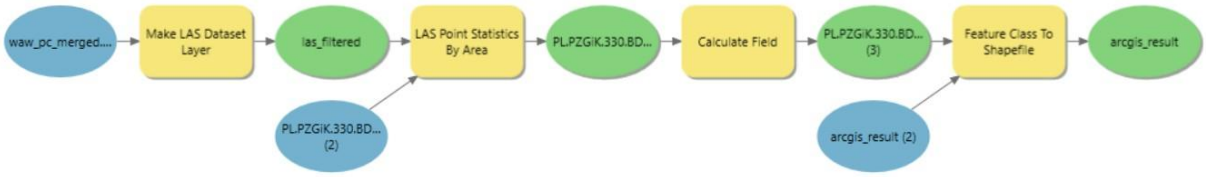
Zauważyć należy również możliwość szybkiego podglądu chmury punktów w zgeneralizowanej wersji, co ułatwia wizualną ocenę danych przed rozpoczęciem analizy (rysunek 62).

Rysunek 62. Wizualizacja fragmentu chmury punktów analizowanej w ramach scenariusza A3 z użyciem oprogramowania QGIS (opracowanie własne)

W odróżnieniu od oprogramowania QGIS, oprogramowanie ArcGIS Pro posiada pełne wsparcie dla operacji zliczania statystyk w poligonach na podstawie chmur punktów. Analiza została przeprowadzona w następujących krokach z użyciem narzędzia ModelBuilder:

1. Wczytanie warstwy LAS i filtracja klasy 6 (narzędzie *Make LAS Dataset Layer*)
2. Wczytanie warstwy BUBD i obliczenie wartości minimalnej i maksymalnej współrzędnej Z w poligonach (narzędzie *LAS Point Statistics By Area*)
3. Obliczenie różnicy pomiędzy maksymalną, a minimalną wartością współrzędnej Z (narzędzie *Calculate Field*)
4. Zapis danych do postaci pliku SHAPEFILE (narzędzie *Feature Class To Shapefile*)

Poszczególne kroki analizy zaimplementowane w ModelBuilder przedstawiono na rysunku 63.



Rysunek 63. Schemat blokowy z modułu *ModelBuilder* oprogramowania ArcGIS Pro utworzony do realizacji scenariusza A3 (opracowanie własne)

Czas obliczeń z użyciem oprogramowania ArcGIS był znacznie większy niż w przypadku oprogramowania QGIS. Średnio, wykonanie całości obliczeń zajęło 1 godzinę 5 minut. Zdecydowanie najdłuższym fragmentem całego przetworzenia było obliczenie statystyk (narzędzie *LAS Points Statistics by Area*). Należy jednak założyć, że etap wczytania chmury punktów i jej filtracja realnie zachodziły również już na tym etapie, gdyż czas wykonania narzędzia *Make Las Dataset Layer* był pomijalnie krótki. Szczegółowy czas trwania poszczególnych etapów w kolejnych próbach przedstawiono w tabeli 22Tabela 22.

Tabela 22. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A3 przeprowadzonego z użyciem oprogramowania ArcGIS Pro z rozbiciem na poszczególne etapy przetwarzania (opracowanie własne)

Próba	Wytworzenie warstwy LAS i filtracja [gg:mm:ss]	Obliczenie statystyk [gg:mm:ss]	Wyliczenie różnicy wysokości [gg:mm:ss]	Zapis do SHP [gg:mm:ss]	Suma [gg:mm:ss]
1	<00:00:01	01:03:57	00:00:12	00:00:53	01:05:02
2	<00:00:01	01:04:20	00:00:13	00:00:52	01:05:26
3	<00:00:01	01:04:42	00:00:13	00:00:52	01:05:47
4	<00:00:01	01:03:43	00:00:12	00:00:51	01:04:46
5	<00:00:01	01:03:35	00:00:14	00:00:52	01:04:40
6	<00:00:01	01:04:14	00:00:13	00:00:52	01:05:19
7	<00:00:01	01:04:02	00:00:14	00:00:53	01:05:08
8	<00:00:01	01:04:36	00:00:12	00:00:52	01:05:40
9	<00:00:01	01:03:18	00:00:13	00:00:52	01:04:22
10	<00:00:01	01:03:51	00:00:13	00:00:52	01:04:56
Średnia	**<00:00:01**	**01:04:02**	**00:00:13**	**00:00:52**	**01:05:07**

W języku Python realizacja scenariusza opartego o chmury punktów odbyła się głównie z użyciem bibliotek laspy i geopandas. Pierwsza z nich posłużyła do wczytania i filtracji chmury punktów, a druga do agregacji danych punktowych w poligonach i obliczenia wynikowych statystyk. Całość kodu, w minimalnej wersji, zajęła 32 linijki. Wykonano następujące operacje:

1. Wczytanie chmury punktów w formacie LAS oraz metadanych (biblioteka laspy)
2. Wczytanie warstwy BUBD w formacie SHP (biblioteka geopandas)
3. Filtracja chmury punktów do punktów reprezentujących klasę 6 (biblioteka laspy)
4. Transformacja współrzędnych punktów odczytanych z pliku LAS do oryginalnych wartości (uwzględnienie skali i przesunięcia z metadanych) (biblioteki laspy oraz NumPy)
5. Transformacja chmury punktów do postaci pandas DataFrame (biblioteka pandas)
6. Utworzenie geometrii na podstawie współrzędnych X i Y i import danych do postaci GeoDataFrame (biblioteka geopandas)
7. Złączenie przestrzenne danych o punktach z danymi o budynkach (biblioteka geopandas)
8. Obliczenie minimalnej i maksymalnej wartości współrzędnej Z dla każdego budynku (biblioteka pandas)
9. Obliczenie różnicy wartości maksymalnej i minimalnej (biblioteka pandas)
10. Zapis wyników do pliku SHP (biblioteka geopandas)

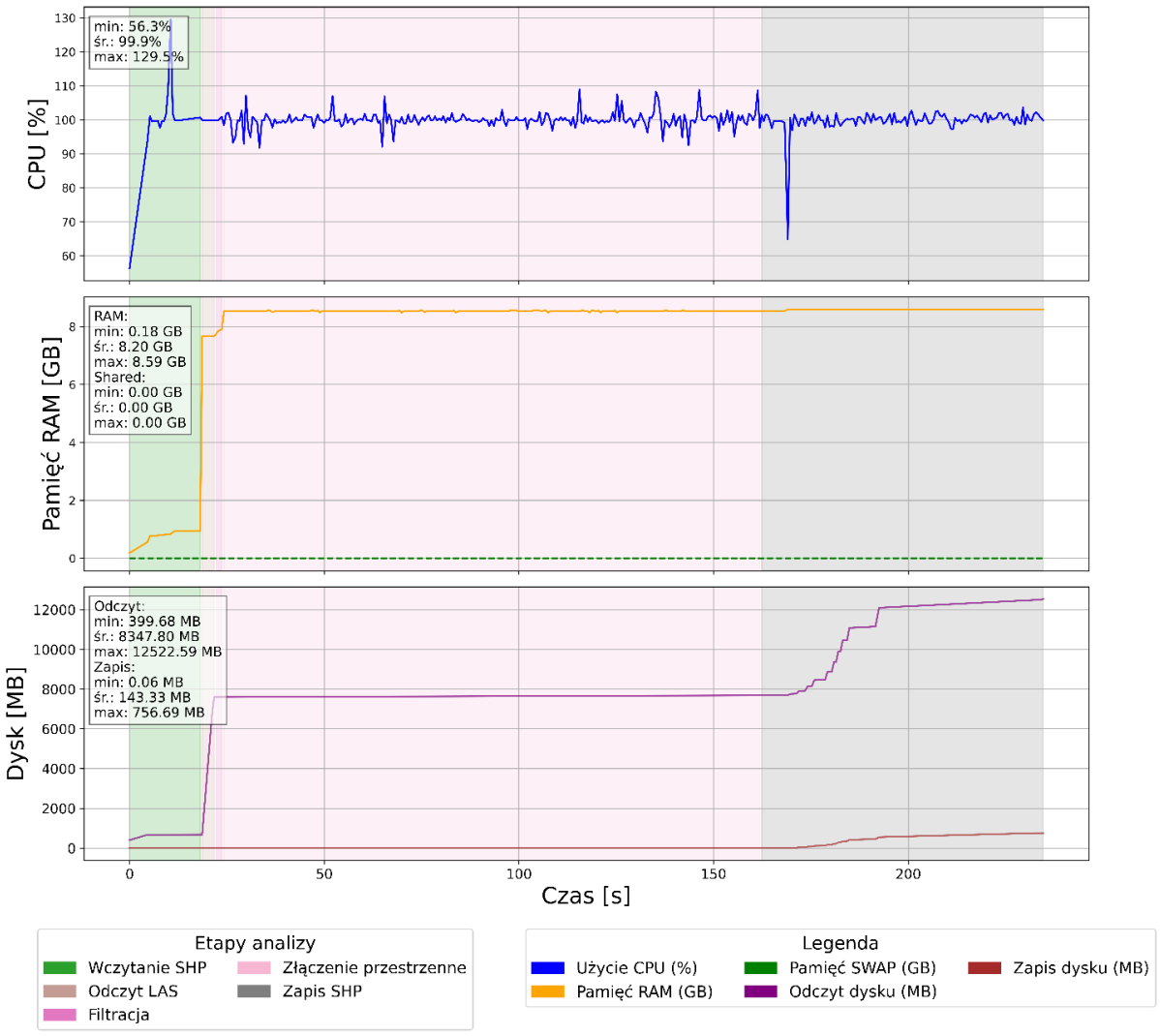
Podobnie, jak w przypadku scenariusza opartego o dane wektorowe, zastosowano w tym przypadku iteracyjne złączenie przestrzenne na podzbiorach całego zbioru w celu uniknięcia wysycenia dostępnej pamięci RAM.

Średni czas potrzebny na wykonanie całej analizy w języku Python wyniósł 4 min 17 sekund. Jest to najkrótszy uzyskany czas spośród wszystkich badanych metod. Najdłuższy fragment obliczeń zajęło złączenie przestrzenne danych (blisko 2 min 30 sek). Szczegóły pomiarów w tym eksperymencie przedstawiono w tabeli 23. Należy zwrócić uwagę, na bardzo krótki czas obliczania statystyk oraz filtracji danych w postaci chmur punktów. Wskazuje to na wysoki stopień optymalizacji bibliotek laspy oraz pandas.

Tabela 23. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A3 zaimplementowanego w języku programowania Python z rozbiciem na poszczególne etapy przetwarzania (opracowanie własne)

Próba	Wczytanie SHP [gg:mm:ss]	Odczyt LAS [gg:mm:ss]	Filtracja [gg:mm:ss]	Złączenie przestrzenne [gg:mm:ss]	Obliczenie statystyk [gg:mm:ss]	Zapis SHP [gg:mm:ss]	Suma [gg:mm:ss]
1	00:00:48	00:00:06	00:00:02	00:02:17	00:00:00	00:01:12	00:04:25
2	00:00:22	00:00:07	00:00:02	00:02:22	00:00:00	00:01:12	00:04:05
3	00:00:23	00:00:07	00:00:02	00:02:39	00:00:00	00:01:13	00:04:25
4	00:00:23	00:00:07	00:00:02	00:02:26	00:00:00	00:01:17	00:04:15
5	00:00:23	00:00:07	00:00:02	00:02:26	00:00:00	00:01:13	00:04:11
6	00:00:23	00:00:08	00:00:02	00:02:27	00:00:00	00:01:12	00:04:12
7	00:00:22	00:00:07	00:00:02	00:02:25	00:00:00	00:01:11	00:04:07
8	00:00:22	00:00:07	00:00:02	00:02:27	00:00:00	00:01:12	00:04:10
9	00:00:24	00:00:09	00:00:03	00:02:58	00:00:00	00:01:15	00:04:50
10	00:00:23	00:00:07	00:00:02	00:02:27	00:00:00	00:01:11	00:04:11
Średnia	**00:00:25**	**00:00:07**	**00:00:02**	**00:02:30**	**00:00:00**	**00:01:13**	**00:04:17**

Analiza zużycia zasobów wskazuje na pełne wykorzystanie pojedynczego wątku procesora przez zdecydowaną większość analizy (rysunek 64). Zauważyć należy również bardzo krótki czas odczytu chmury punktów z dysku do pamięci operacyjnej. W przeciągu ok. 7 sekund cała chmura została wczytania zapełniając ponad 8GB pamięci RAM. Pamięć ta, po wczytaniu danych SHP i chmury punktów nie jest już bardziej zajmowana, zachowując duży zapas do założonych 32GB całej pamięci operacyjnej.



Rysunek 64. Wykres wykorzystania poszczególnych zasobów (CPU, RAM, Dysk) w czasie trwania obliczeń dla scenariusza A3 w języku programowania Python (opracowanie własne)

Podobnie, jak w przypadku scenariusza dotyczącego danych wektorowych, przed rozpoczęciem właściwych obliczeń w Spark należało przetworzyć wejściowe dane tak, aby były dostępne na przestrzeni HDFS dla wszystkich maszyn *worker*. W tym celu dokonano transformacji chmury punktów z postaci pliku LAS do postaci GeoMesa-FS. Operacja ta zajęła średnio 31min 25s. Przy jej realizacji konieczne było wprowadzenie zmiany do wejściowego formatu danych. Chmura punktów w postaci LAS przechowuje część danych w formacie liczby całkowitej bez znaku (*unsigned integer*). Apache Spark nie obsługuje tego typu danych, więc konieczna była konwersja do formatu liczby całkowitej ze znakiem (*signed integer*). Kolejnym krokiem było transformacja danych z układu PL-2000 do układu WGS 1984 obsługiwanego przez GeoMesa-FS. W analogiczny sposób przygotowano warstwę BUBD z geometrią budynków. Sama analiza została przeprowadzona w następujących krokach:

1. Odczyt danych GeoMesa-FS z chmurami punktów i budynkami
2. Filtracja punktów tylko do tych znajdujących się w klasie 6
3. Złączenie przestrzenne obu zbiorów z użyciem metody opartej o Pandas UDF
4. Obliczenie wartości najmniejszej i największej współrzędnej Z w poligonach
5. Obliczenie różnicy pomiędzy najmniejszą i największą współrzędną
6. Załadowanie wyniku na maszynę driver i zapis do pliku SHP

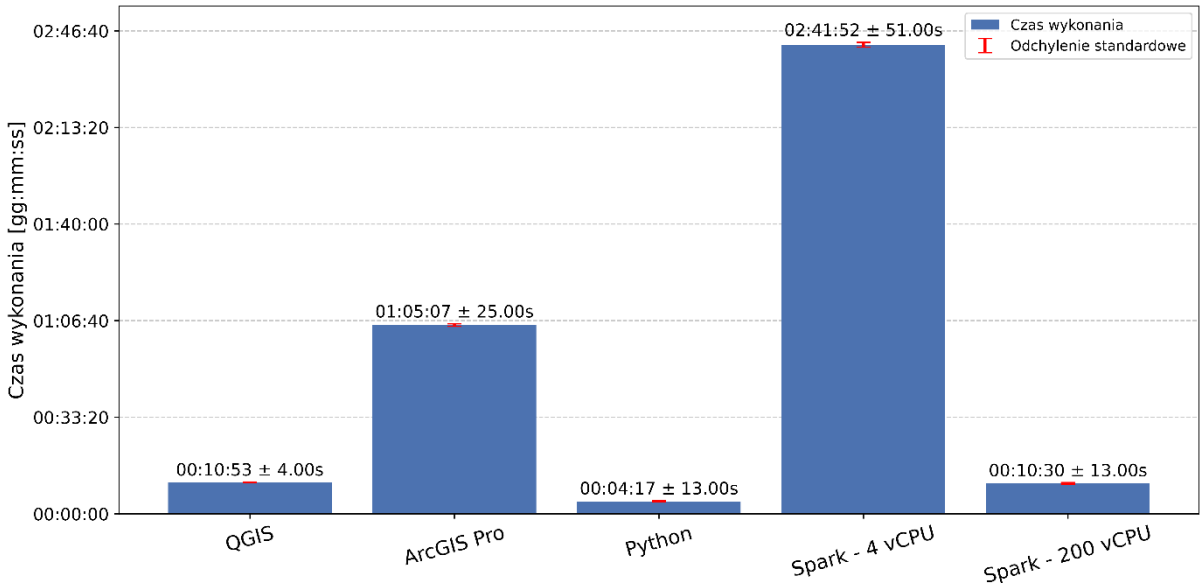
Czas obliczeń z użyciem Apache Spark wyniósł ponad 10 minut z wykorzystaniem 200 vCPU oraz 2 godziny 42 minuty dla 4 vCPU (szczegóły pomiarów w poszczególnych próbach zaprezentowano w tabeli 24). W porównaniu do innych metod zauważyć można znacznie niższą wydajność.

Tabela 24. Podsumowanie czasu trwania kolejnych prób realizacji scenariusza A3 zaimplementowanego w środowisku Spark z rozbiciem na wykorzystaną moc klastra obliczeniowego (opracowanie własne)

Próba	Spark (200 vCPU) [gg:mm:ss]	Spark (4 vCPU) [gg:mm:ss]
1	00:10:39	02:41:33
2	00:10:14	02:42:47
3	00:10:16	02:40:30
4	00:10:57	02:43:32
5	00:10:32	02:42:04
6	00:10:24	02:41:55
7	00:10:19	02:40:45
8	00:10:42	02:41:58
9	00:10:39	02:42:06
10	00:10:21	02:41:29
Średnia	00:10:30	02:41:51

W tym przypadku uzyskanie wyższej wydajności z pewnością byłoby możliwe przy zastosowaniu optymalizacji partycjonowania czy optymalizacji filtrowania danych. Należy jednak zauważyć, że już w domyślnej wersji konieczne było wprowadzenie szeregu dodatkowych operacji związanych z przygotowaniem danych. Kod w środowisku Spark zajął, w minimalnej wersji, 57 linijek podstawowego skryptu oraz 86 linijek skryptu konwertującego z formatu LAS do formatu GeoMesa-FS.

Spośród wszystkich analizowanych metod najkrótszy czas obliczeń został osiągnięty z użyciem języka Python. Bardzo dobry wynik udało się osiągnąć również z wykorzystaniem oprogramowania QGIS, które pomimo braku bezpośredniego wsparcia dla analiz obszarowych chmur punktów pozwoliło na realizację zadania w czasie ok. 11 minut. Jest to czas bliski temu, który był potrzebny na wykonanie analizy na klastrze Spark z użyciem 200 vCPU. Klaster wyposażony w pojedynczą maszynę *worker* potrzebował 2 godzin 42 minut na zakończenie obliczeń. Oprogramowanie ArcGIS Pro cechowało się najmniejszym stopniem skomplikowania przeprowadzenia założonej analizy, lecz jej realizacja wymagała średnio ponad 1 godzinę 5 minut. Zestawienie wydajności poszczególnych rozwiązań w tym zadaniu przedstawiono na rysunku 65.

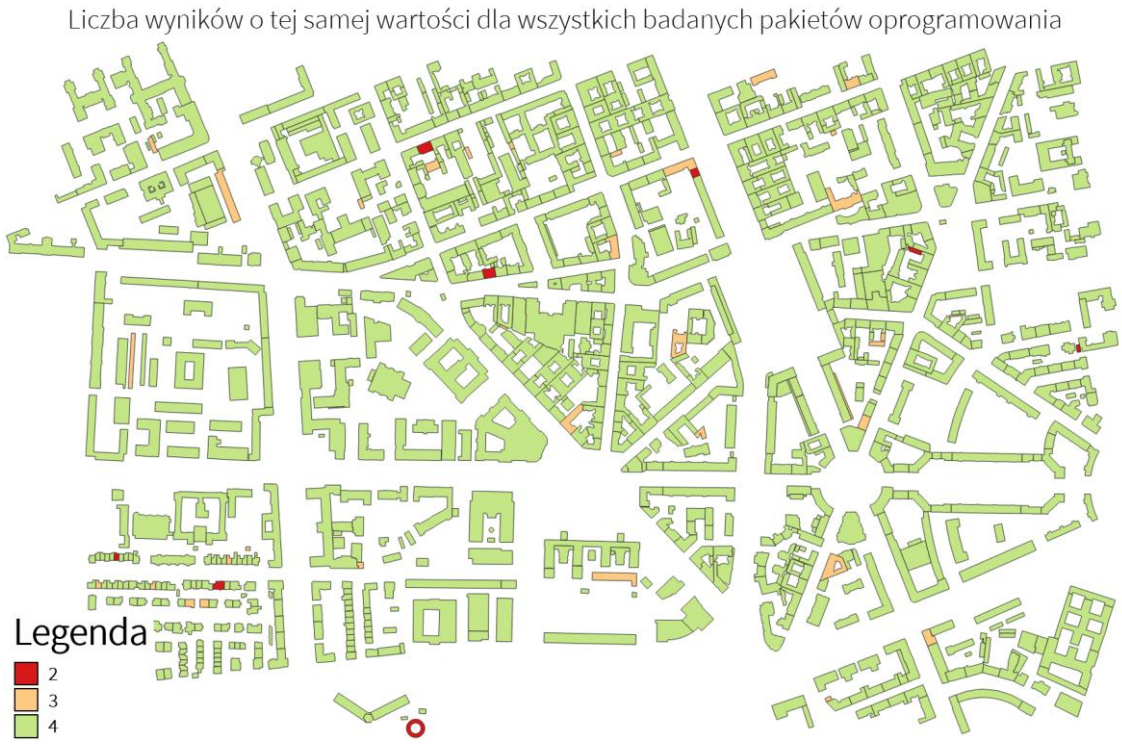


Rysunek 65. Porównanie średniego czasu wykonania scenariusza A3 w zależności od zastosowanego oprogramowania (opracowanie własne)

W wyniku wszystkich czterech podejść do analizy do większości z 986 obiektów przypisana została ta sama wysokość. Jednak, w przypadku 34 obiektów trzy metody dały ten sam wynik, a w przypadku 8 wyników jedynie dwie z nich. We wszystkich przypadkach, w których wynik był wspólny dla dwóch wartości, oprogramowanie QGIS i ArcGIS przypisywało te same wyniki, a skrypt Python oraz Spark przypisywały identyczną parę wartości. Może to być więc spowodowane inną zasadą łączenia przestrzennego użytą w skryptach Python/Spark oraz QGIS/ArcGIS. Dla Python/Spark zastosowano operator *Within*, gdzie dla QGIS/ArcGIS zasady złączenia mogą być inne, tak jak w przypadku scenariusza dotyczącego danych wektorowych. W pozostałych 34. przypadkach różnic, żadna nie występuje pomiędzy ArcGIS, a QGIS, oba te

wyniki są identyczne. W przypadku 8 różnic, jest ona spowodowana innym wynikiem w skrypcie Python (w Spark wynik jest identyczny jak w ArcGIS/QGIS). W dwóch przypadkach różnica ta przekracza metr, a jej mediana wynosi 11.5 cm.

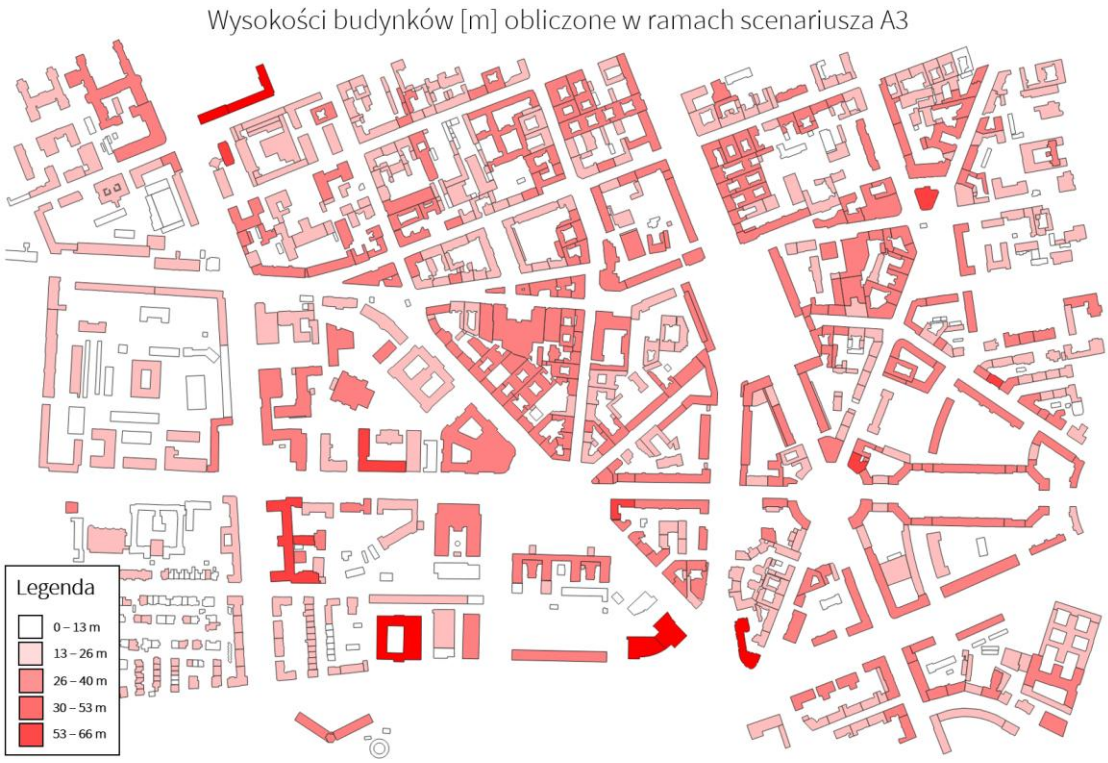
Pozostałe 26 różnic dotyczy skryptu opartego o Spark. W wynikach zaobserwować można jeden wynik znacznie odstający (3.47 m), a mediana różnic wynosi 2 cm. Różnice w przypadku skryptu Spark mogą wynikać z procesu transformacji współrzędnych pomiędzy układami i możliwej utraty precyzji na etapie zaokrąglenia wartości. W przypadku skryptów Python oraz Spark znaczne różnice mają wyłącznie znak ujemny, co sugeruje, że część punktów analizowanych w QGIS/ArcGIS nie została uwzględniona w obrębie budynku. Na rysunku 65 przedstawiono obiekty, gdzie wystąpiły różnice w wynikach pomiędzy poszczególnymi podejściami. Jak widać różnice występowały równomiernie na całym obszarze opracowania, co może sugerować, że ich źródło związane jest ze sposobem agregacji punktów w obszarach lub precyzją obliczeń, a nie czynnikiem związanym z rozkładem przestrzennym (np. mechanizmami kafelkowania).



Rysunek 66. Wizualizacja różnic w wynikach realizacji scenariusza A3 pomiędzy poszczególnymi pakietami oprogramowania (opracowanie własne)

Scenariusz dotyczący chmur punktów został przeprowadzony z użyciem wszystkich czterech pakietów oprogramowania. W każdym przypadku udało się uzyskać oczekiwany wynik w tej samej formie. Operacje na chmurach punktów są wspierane przez wszystkie analizowane

środowiska, jednak należy zauważyć, że zarówno w oprogramowaniu QGIS, jak i ArcGIS Pro część narzędzi dobrze znanych z analizy danych wektorowych bądź rastrowych nie jest bezpośrednio dostępna w przypadku chmur punktów i wymaga zastosowania dodatkowych operacji. W przypadku języka Python czy środowiska Apache Spark chmury punktów interpretowane są dokładnie w taki sam sposób jak punktowe dane wektorowe, co ułatwia implementację. Jednak w środowisku Python możliwe jest bezpośrednie wykorzystanie bibliotek takich jak laspy pozwalających na odczyt i analizę danych w postaci plików LAS. Środowisko SBD CENAGIS oparte o Apache Spark nie zapewnia metod zdolnych do bezpośredniego odczytu danych w formacie LAS z systemu plików HDFS. Na rysunku 67 przedstawiono wynik opracowania scenariusza A3 w formie wizualizacji kartograficznej.



Rysunek 67. Wynik realizacji scenariusza A3 w formie wizualizacji kartograficznej prezentującej wysokości budynków w zakresie opracowania (opracowanie własne)

9.4. Podsumowanie badań porównawczych

Wszystkie trzy założone scenariusze badawcze udało się zrealizować z użyciem czterech podstawowych pakietów oprogramowania GIS: QGIS, ArcGIS Pro, Python oraz SBD CENAGIS. Analiza statystyczna wyników we wszystkich scenariuszach wykazała wysoką powtarzalność pomiarów pomiędzy próbami, co wskazuje niewystępowanie podczas analiz błędów

przypadkowych oraz brak istotnego nieprzewidywalnego wpływu czynników zewnętrznych takich jak obciążenie klastra innymi obliczeniami czy jego awarie.

Pomimo stosunkowo prostych założeń co do przetworzeń założonych w ramach scenariuszy na tym etapie napotkano wiele problemów specyficznych dla poszczególnych środowisk.

Narzędzia do graficznego budowania analiz takie jak Model Builder w oprogramowaniu QGIS i ArcGIS pozwoliły na stosunkowo wygodne i powtarzalne przeprowadzenie analiz przestrzennych. Oba opisywane narzędzia zapisywały też czas wykonania poszczególnych kroków przetworzenia, co ułatwiło analizę ich wydajności. Narzędzia w ramach obu pakietów oprogramowania nie pozwalały jednak na bardzo dokładną kontrolę nad procesem wykonania poszczególnych operacji. Przykładowo, narzędzie „*LAS Points Statistic By Area*" nie pozwala na wybór sposobu złączenia przestrzennego zbiorów, umożliwiając jedynie wybór, jakie statystyki mają być policzone. Oczywiście istnieją możliwości obejścia tego ograniczenia poprzez zamianę danych LAS do postaci wektorowej bądź rastrowej, jednak wymaga to już wykonania dodatkowych operacji.

Wszystkie trzy analizy zostały wykonane najszybciej z użyciem języka Python (przy założeniu wykorzystania pojedynczej maszyny obliczeniowej). Wykorzystanie języka programowania pozwoliło też na największą kontrolę nad realizowanymi przetworzeniami, a bogaty zbiór bibliotek wspierających obliczenia przestrzenne zapewnił stosunkowo prostą i intuicyjną implementację scenariuszy badawczych bez konieczności wdrażania dedykowanych rozwiązań programistycznych. Należy również podkreślić kluczowy wpływ na wydajność wersji bibliotek w przypadku korzystania z języka Python. Dobrym przykładem tego wpływu może być zastosowanie biblioteki shapely w wersji 2 w porównaniu do wersji wcześniejszych. Zastosowana w przypadku środowiska Python wersja 2 biblioteki shapely wspiera zwektoryzowane operacje na geometriach optymalizowane przez bibliotekę NumPy. W związku z tym operacje wymagające złączeń przestrzennych wykonanych w bibliotece geopandas (która oparta jest wewnętrznie o shapely) są od wersji 2 znacznie szybsze. Przykładowo, realizacja scenariusza A1 na środowisku opartym o niższe wersje bibliotek zajęło średnio 31 minut 55 sekund zamiast 5 minut 21 sekund. Środowisko Python pozwoliło też na bardzo dokładne śledzenie aktualnego zużycia zasobów, nie tylko w rozbiciu na czas trwania poszczególnych etapów, ale także sekunda po sekundzie możliwy był zapis wykorzystania procesora, pamięci

RAM czy dysków twardych. Jest to wykonalne w przypadku środowiska desktop GIS, ale wymaga opracowania dedykowanego rozwiązania monitorującego, a także może być obarczone błędem związanym z wyznaczeniem czasu rozpoczęcia obliczeń bądź specyfiką wykorzystania zasobów przez to oprogramowanie.

Wykorzystanie środowiska SBD CENAGIS opartego o Apache Spark do realizacji wszystkich trzech scenariuszy było możliwe, ale wymagało dodatkowych kroków związanych z przygotowaniem danych, zapisem ich w rozproszonym systemie plików HDFS oraz uwzględnieniem zachodzenia obliczeń rozproszonych. Szczególnie w przypadku analizy rastrowej konieczne było wprowadzenie dedykowanych rozwiązań zapewniających poprawność wyniku na granicy kafelków rastra podzielonego z użyciem biblioteki RasterFrames. We wszystkich przypadkach operacje rozproszone z użyciem pojedynczej maszyny *worker* o analogicznych parametrach zakończyły się wolniejszym czasem obliczeń niż w przypadku języka Python oraz wolniejszym bądź zbliżonym czasem w przypadku oprogramowania desktop GIS. Wykorzystanie większych mocy obliczeniowych pozwoliło na uzyskanie najszybszego czasu przetworzeń w przypadku Scenariusza 1, ale w innych przypadkach czas obliczeń nadal był dłuższy niż ten osiągnięty przez standardowe narzędzia. W przypadku narzędzi SBD należy również podkreślić wpływ wersji bibliotek na wydajność obliczeń. Środowisko SBD CENAGIS jest złożonym produktem informatycznym zawierającym wiele dedykowanych rozwiązań nabudowanych na istniejące biblioteki *open source* takie jak GeoMesa czy RasterFrames. W takich warunkach aktualizacja poszczególnych komponentów środowiska jest dużo bardziej czasochłonna i wymaga udziału doświadczonych i znających środowisko specjalistów. Z perspektywy użytkownika oznacza to dużo wolniejsze tempo aktualizacji bibliotek działających w ramach maszyn *worker* środowiska Apache Spark. Dotyczy to również bibliotek języka Python takich jak geopandas czy shapely, których wpływ na wydajność środowiska będzie ujawniał się w przypadku zastosowania metod opartych o funkcje pandas UDF. Aktualnie w środowisku CENAGIS biblioteka shapely jest w wersji poniżej 2 co oznacza, że opisywane powyżej optymalizacje nie mogły być zastosowane w przypadku złączeń przestrzennych opartych o funkcje pandas UDF. Sugeruje to też duży potencjał zwiększenia wydajności środowiska w przypadku aktualizacji jego komponentów zarówno tych związanych z przetwarzaniem SBD, jak i klasycznych bibliotek języka Python.

Dokładne śledzenie zużycia zasobów przez środowisko Apache Spark jest trudne z kilku powodów. Po pierwsze operacje wykonane z użyciem Apache Spark są ewaluowane „leniwie" po optymalizacji planu wykonania. Nie da się więc dokładnie rozgraniczyć czasu wykonania poszczególnych operacji – wszystkie wykonują się razem po inicjalizacji obliczeń poprzez wywołanie metody wymuszającej prezentacje wyników (np. *.show()*[70] albo *.toPandas()*[71]). Co więcej, obliczenia wykonują się realnie na wielu fizycznych serwerach na raz wykorzystując ich moce obliczeniowe, ale też częściowo na maszynie *driver*. Dodatkowy czas obliczeń związany jest również z tworzeniem planu wykonania, listowaniem plików na HDFS czy przenoszeniem danych pomiędzy maszynami. System dokładnie śledzący wydajność operacji wykonanych z użyciem Apache Spark musiałby zostać wykonany w sposób dedykowany dla środowiska CENAGIS.

Zastosowanie środowiska SBD do standardowej wielkości danych przestrzennych nie ma więc uzasadnienia. W takich przypadkach lepszym wyborem może być implementacja analiz w języku Python, gdyż zapewnia to większą dowolność w optymalizacji rozwiązań w porównaniu z narzędziami desktop GIS, w stosunkowo prosty sposób pozwala na dokładne śledzenie zużycia zasobów, a także skraca ścieżkę do wdrożenia przetworzeń w sposób rozproszony w środowisku SBD, ze względu na fakt, iż to środowisko również wykorzystuje język Python.

Zauważyć należy jednak pewne przewagi środowiska SBD nawet przy znajomości opisanych powyżej ograniczeń i dodatkowych wymogów. W przypadku konieczności szybkiego skalowania obliczeń (np. konieczność wykonania tych obliczeń dla znacznie większego obszaru), po implementacji analizy GIS w środowisku SBD będzie to możliwe bezpośrednio poprzez zmianę jednego parametru. Analizy oparte o język Python będą wymagać zastosowania dodatkowych rozwiązań bądź zmian właśnie z uwzględnieniem obliczeń rozproszonych (np. wykorzystanie Dask lub przepisanie kodu na pySpark). Efekt skali może zniwelować gorszą optymalizację poszczególnych metod, czego przykładem może być scenariusz A1.

[70] Komenda wyświetlająca małą próbkę wyników w postaci tabeli.
[71] Komenda konwertująca zbiór danych w postaci Spark DataFrame do postaci pandas DataFrame przy jednoczesnym przeniesieniu całości wyników na maszynę driver.

Z pewnością każdy z przedstawionych scenariuszy badawczych w każdym z narzędzi da się dalej zoptymalizować przyspieszając wykonanie obliczeń. Należy jednak wziąć pod uwagę czas potrzebny do wdrożenia optymalizacji i fakt, że niektóre narzędzia zawsze będą cechować się charakterystycznymi ograniczeniami. Przeprowadzone analizy wskazują, że dla zbadanych przypadków możliwe jest osiągnięcie wyższej wydajności w podobnym czasie wykorzystując język programowania Python w porównaniu do narzędzi desktop GIS. Wykorzystanie środowiska SBD wymaga jednak specyficznych warunków, żeby jego wykorzystanie przyniosło wartość z perspektywy specjalisty GIS. Przede wszystkim wykorzystanie tych narzędzi może być uzasadnione w przypadku, gdy optymalizacje związane z wykorzystaniem pojedynczej maszyny obliczeniowej nie pozwalają na poprawne wykonanie całości analizy, albo gdy z góry założona ilość danych nie pozwoli na wykonanie obliczeń w założonym czasie. Nawet w takim przypadku uzasadnione jest wykonanie analizy np. w języku Python w celu późniejszej konwersji do systemu SBD CENAGIS (albo innych, opartych np. o Dask) na mniejszym zbiorze danych i analiza czy istnieje zasadność wdrożenia wersji SBD przetworzeń.

Przedstawione powyżej rozważania potwierdzają hipotezę tego etapu badań: Nie dla wszystkich analiz przestrzennych uzasadnione jest wdrożenie narzędzi SBD.

Dla zbiorów danych, których opracowanie możliwe jest w ramach pojedynczej stacji roboczej we wszystkich przypadkach wykorzystanie klasycznych metod okazało się być bardziej wydajne (przy założeniu tej samej mocy obliczeniowej). Uzasadnienie wdrożenia narzędzi SBD można więc rozważać w przypadkach, gdy wielkość zbioru wymusza wstępny podział danych na mniejsze podzbiory ze względu na przepełnianie się zasobów obliczeniowych bądź przestrzeni dyskowej. Drugim uzasadnieniem wdrożenia narzędzi SBD może być konieczność uzyskania wysokiej, skalowalnej wydajności bez względu na koszty związane z dostępem do mocy obliczeniowych. W tym wypadku, dla części analiz możliwe jest „siłowe" zwiększenie prędkości obliczeń ponad możliwości pojedynczej maszyny (tak jak to miało miejsce w przypadku Scenariusza 1). Należy przy tym pamiętać, że bez czasochłonnych optymalizacji i testów może to nie być możliwe, w szczególności w przypadku bardziej złożonych analiz wykorzystujących algorytmy niezaimplementowane w wydajny sposób w Spark bądź bibliotekach uzupełniających.

10. Badania wydajności środowiska SBD

Ten rozdział opisuje scenariusze badawcze oraz wyniki ich realizacji w ramach drugiego etapu badań, który skupił się na analizie wydajności samego środowiska SBD CENAGIS.

10.1. Scenariusz B1 – podstawowe zapytania na danych FCD

Pierwszy scenariusz skupia się na badaniu podstawowych zapytań na danych FCD wraz z ich importem do środowiska SBD. Jest to krok konieczny do wydajnego przeprowadzenia dowolnych innych analiz w tym środowisku. Możliwe jest użycie w analizach formatu wejściowego danych FCD (w tym przypadku CSV), jednak nie ma możliwości wykorzystania wtedy przewag indeksowania przestrzennego i czasowego, dane zajmują więcej miejsca, a ich odczyt będzie mniej wydajny względem formatów zoptymalizowanych pod przetwarzanie SBD np. Parquet. Zastosowanie formatu Parquet pozwala na wykorzystanie optymalizacji odczytu danych związanych nie tylko z indeksowaniem GeoMesa, ale także z samą kolumnową strukturą tych plików. W przypadku filtrowania bądź selekcji kolumn możliwe jest zastosowanie mechanizmu filtrowania *push-down*, który jest realizowany na etapie odczytu danych (Ivanov i Pergolesi, 2020). Wykorzystuje on metadane każdej partycji pliku Parquet w celu ograniczenia odczytu danych tylko do tych partycji, które mogą zawierać poszukiwane rekordy.

W scenariuszu B1 połączono dane o rzeczywistych średnich prędkościach pojazdów na drodze z informacją geometryczną (sieć drogowa OpenStreetMaps), a następnie całość zapisano do formatu Parquet GeoMesa-FS z zastosowaniem różnych parametrów indeksowania przestrzennego, czasowego oraz kompresji danych. Wykonano także próbę zapisu tego zbioru do bazy danych Accumulo. Pierwsze testy wykazały bardzo niską wydajność bazy danych Accumulo w badanym systemie i nie było możliwe poprawne zapisane analizowanej próbki danych. Niska wydajność bazy Accumulo związana jest z faktem, iż za jej działanie odpowiadają osobne jednostki obliczeniowe, których skalowanie nie jest zautomatyzowane w środowisku SBD CENAGIS. W związku z tym operacje zapisu i odczytu z bazy stają się wąskim gardłem, uniemożliwiając wykonanie analiz dla dużych zbiorów danych. Dlatego dalsze badanie przeprowadzono z użyciem formatu GeoMesa-FS.

Oryginalne dane zapisano w systemie plików HDFS, co umożliwiło wydajne wykorzystanie przetwarzania rozproszonego w oparciu o Apache Spark, a następnie przeprowadzono serię badań wydajnościowych dotyczących odczytu danych oraz prostych zapytań czasowych, przestrzennych i czasowo-przestrzennych. W tabeli 25 przedstawiono badane parametry zapisu do formatu GeoMesa-FS oraz przyjęte do analizy wartości.

Tabela 25. Podsumowanie wartości przyjętych w parametrach badanych w ramach scenariusza B1 - importu danych do postaci pliku Parquet w formacie GeoMesa-FS (opracowanie własne)

Badany parametr	Przyjęte wartości
Kompresja	Brak, Snappy[72], Gzip[73]
Wielkość indeksu przestrzennego	2 bity, 4 bity, 8 bitów, 12 bitów, 16 bitów
Granulacja indeksu czasowego	Brak indeksu, dzienna, miesięczna

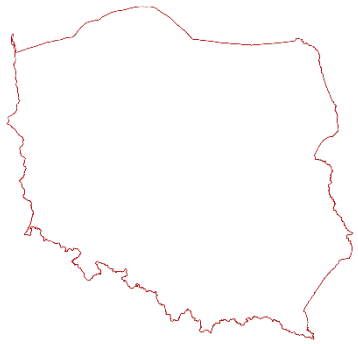
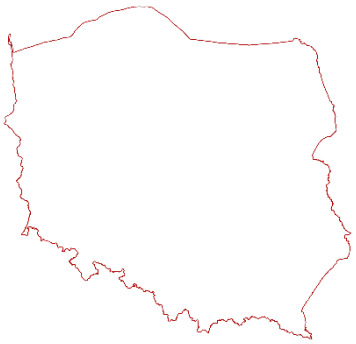
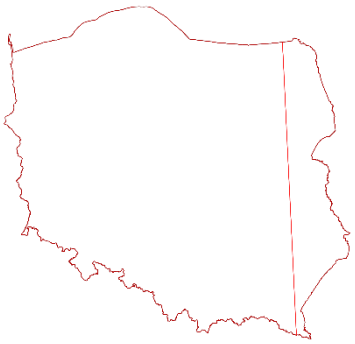
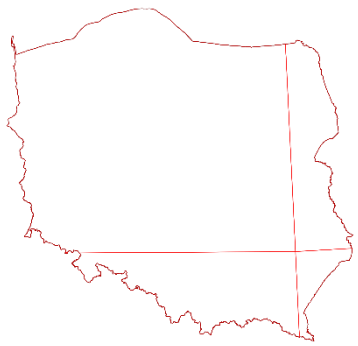
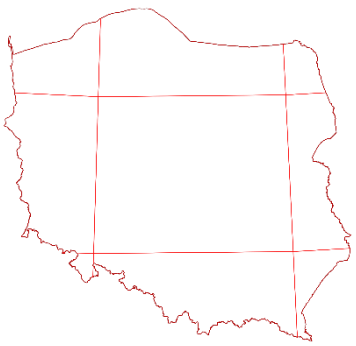
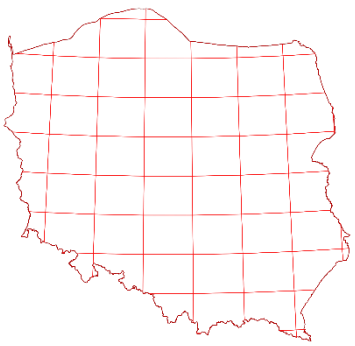
Indeks przestrzenny w GeoMesa koduje położenie obiektów w postaci jednowymiarowego klucza. Liczba bitów indeksu przestrzennego determinuje jego rozdzielczość — większa liczba bitów pozwala na dokładniejsze odwzorowanie relacji przestrzennych, lecz prowadzi do większej liczby partycji i potencjalnie wyższego kosztu zarządzania indeksem. Mniejsza liczba bitów skutkuje mniejszą liczbą kafelków podziału przestrzennego. Indeks czasowy w GeoMesa umożliwia logiczną organizację danych według wymiaru czasowego poprzez grupowanie rekordów w przedziały o określonej granulacji (np. dziennej lub miesięcznej). Zastosowanie indeksu czasowego pozwala na efektywne ograniczenie zakresu przeszukiwanych danych w zapytaniach uwzględniających czas, kosztem potencjalnego zwiększenia liczby partycji , a w efekcie liczby plików zapisywanych na dysku. Brak indeksu czasowego eliminuje ten narzut, lecz prowadzi do konieczności przeszukiwania większego wolumenu danych przy analizach wykorzystujących kolumnę z informacją o czasie.

W tabeli 26 przedstawiono rozkład kafelków podziału, które przypadają na obszar Polski w zależności od zastosowanego rozmiaru przestrzennego kafelka. Indeksowanie GeoMesa-FS realizowane jest w skali całego świata, w związku z czym indeksy o większym rozmiarze

[72] Bezstratny, otwarty format kompresji danych skoncentrowany na szybkości działania opracowany przez firmę Google (kod źródłowy dostępny pod adresem: https://github.com/google/snappy)
[73] Bezstratny format kompresji danych oparty o algorytm Deflate (Deutsch, 1996)

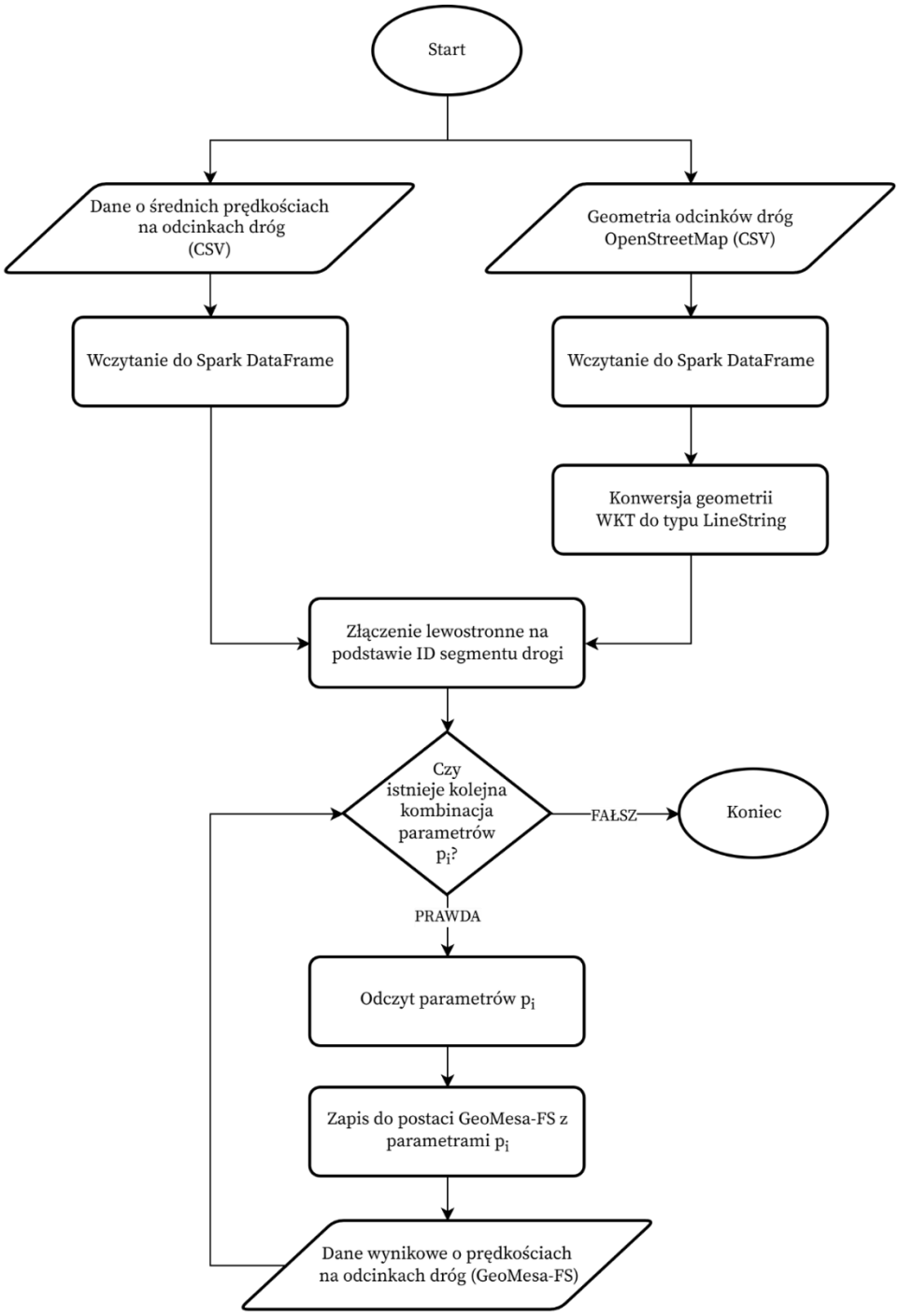
przestrzennym kafelka przypisują dla obszaru Polski jeden obszar, co uniemożliwia uzyskanie jakichkolwiek korzyści na etapie odczytu tak zaindeksowanych danych. Testy indeksów tej wielkości mają jednak na celu zbadanie działania samego mechanizmu indeksowania i jego wpływu na ostateczny sposób zapisu danych, a także wydajność ich odczytu i analizy.

Tabela 26. Wizualizacja liczby kafelków indeksu przypadających na terytorium Polski w zależności od rozmiaru przestrzennego kafelka podczas zapisu danych w formacie GeoMesa-FS (opracowanie własne)

Indeks 2 bit (1 kafelek indeksu)	Indeks 4 bit (1 kafelek indeksu)	Indeks 8 bit (2 kafelki indeksu)
		
Indeks 10 bit (4 kafelki indeksu)	**Indeks 12 bit (9 kafelków indeksu)**	**Indeks 16 bit (61 kafelków indeksu)**
		

Proces zapisu danych do formatu Parquet połączono z integracją danych o geometrii odcinków dróg, której brakowało w danych wejściowych. Proces ten zrealizowano zgodnie ze schematem przedstawionym na rysunku 68. Dodanie informacji o geometrii do każdego rekordu prowadzi do zwiększenia redundancji danych. W praktyce nie stanowi to jednak istotnego narzutu pamięciowego, ponieważ kolumnowy format plików Parquet umożliwia bardzo efektywną kompresję powtarzalnych wartości. Co istotniejsze, w paradygmacie przetwarzania *big data* redundancja danych jest często celowo akceptowana, a nawet pożądana, gdyż pozwala

ograniczyć liczbę kosztownych operacji obliczeniowych, takich jak złączenia przestrzenne czy odwołania do zewnętrznych struktur danych. Przeniesienie części informacji do poziomu pojedynczego rekordu umożliwia wydajne przetwarzanie danych w obrębie partycji, redukuje konieczność komunikacji pomiędzy węzłami klastra oraz poprawia skalowalność i deterministyczność czasu wykonania analiz.



Rysunek 68. Schemat blokowy procesu importu danych FCD w ramach scenariusza B1. p_i oznacza i-tą kombinację parametrów zapisu danych (indeksowanie oraz kompresja), analizowaną w ramach eksperymentu (opracowanie własne)

Tak zapisane dane zostały następnie poddane serii eksperymentów mających na celu zbadanie czasu przetworzeń w zależności od zastosowanych parametrów zapisu:

- Czas odczytu danych poprzez wymuszenie odczytu całego zbioru z HDFS poprzez użycie polecenia zliczania obiektów *count()*
- Czas wykonania zapytania przestrzennego polegającego na znalezieniu rekordów danych wewnątrz prostokąta ograniczającego Warszawę i okolice:

dataframe.filter("st_contains(st_makeBBOX(20.77,52.10,21.25,52.39), geom)")

- Czas wykonania zapytania filtrującego dane zarejestrowane w styczniu 2019 i 2020:

dataframe.filter((F.col("dtg") >= F.lit("2019-01-01 00:00:00.000 ") .cast(TimestampType())) &

(F.col("dtg") < F.lit("2019-02-01 00:00:00.000").cast(TimestampType())) |

(F.col("dtg") >= F.lit("2020-01-01 00:00:00.000").cast(TimestampType())) &

(F.col("dtg") < F.lit("2020-02-01 00:00:00.000").cast(TimestampType()))

- Czas wykonania zapytania wyszukującego rekordy wewnątrz podanego prostokąta ograniczającego zarejestrowane w styczniu 2019 i 2020:

dataframe.filter(f"st_contains(st_makeBBOX(20.77,52.10,21.25,52.39), geom)").filter((F.col("dtg") >= F.lit("2019-01-01 00:00:00.000").cast(TimestampType())) &

(F.col("dtg") < F.lit("2019-02-01 00:00:00.000").cast(TimestampType())) |

(F.col("dtg") >= F.lit("2020-01-01 00:00:00.000").cast(TimestampType())) &

(F.col("dtg") < F.lit("2020-02-01 00:00:00.000").cast(TimestampType()))

W tym scenariuszu wszystkie eksperymenty przeprowadzono z użyciem następujących parametrów klastra: 32 maszyny *worker* wyposażone w8 vCPU i 24 GB RAM każda.

Do badań nad importem danych wykorzystano 10% całego zbioru o średnich prędkościach pojazdów na odcinkach dróg. Taki fragment zbioru przekłada się na 1392 GB danych w postaci plików CSV, przechowujących 37.9 miliarda rekordów.

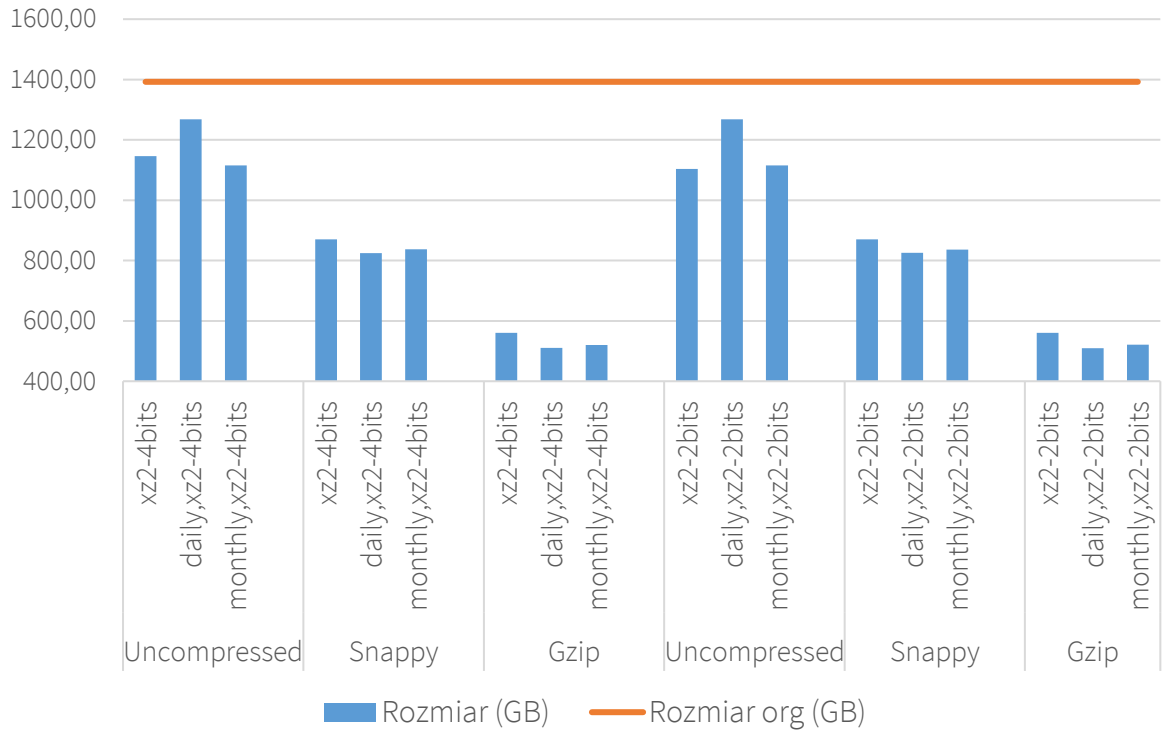
W pierwszej iteracji zbadano wpływ na czas przetwarzania wybranego sposobu kompresji (brak kompresji, kompresja Snappy oraz kompresja Gzip) oraz rodzaju indeksu dla kolumny z datą (brak indeksowania, indeksowania miesięczne oraz indeksowanie dzienne). W każdym z badanych przypadków zastosowano indeks przestrzenny 4 oraz 2 bitowy. Następnie zmierzono czas wykonania operacji zapisu, liczbę utworzonych plików oraz zajmowane miejsce w porównaniu do analogicznego podzbioru danych w oryginalnej formie (plik CSV bez dołączonej informacji o geometrii).

Proces importu składał się z następujących kroków:

1. Pobranie plików CSV z danymi o prędkościach pojazdów na odcinkach dróg z bazy danych uzyskanej z bazy Yanosik na pojedynczą maszynę i zapis ich na dysku współdzielonym Ceph
2. Pobranie pliku CSV z siecią drogową z bazy OSM na pojedynczą maszynę i zapis na dysku współdzielonym Ceph
3. Kopia wszystkich plików CSV w niezmienionej formie do systemu plików HDFS
4. Odczyt plików CSV z danymi o prędkościach na odcinkach dróg w postaci Spark DataFrame
5. Odczyt pliku CSV z siecią drogową w postaci Spark DataFrame
6. Przetworzenie geometrii w formie *Well-Know Text* (WKT) do obiektu typu *LineString* biblioteki GeoMesa dla obiektu DataFrame z siecią drogową
7. Wykonanie złączenia lewostronnego danych o prędkości z geometrią sieci drogowej na podstawie klucza ID segmentu drogi
8. Zapis połączonych danych w postaci GeoMesa-FS z zadanym sposobem kompresji oraz metodą indeksowania

Na etapie zapisu wykorzystano domyślną liczbę partycji utworzoną przez Spark po wykonaniu powyższych kroków (20 480). Na rysunku 69 pokazano rozmiar analizowanej próbki danych po zapisie w systemie plików HDFS (bez replikacji) w zależności od zastosowanych parametrów zapisu. Wszystkie badane wersje zbioru zapisane na HDFS cechują się mniejszym rozmiarem niż dane w formacie wejściowym (CSV). W przypadku kompresji GZIP różnica ta jest ponad dwukrotna. Zgodnie z oczekiwaniami główny wpływ na wielkość danych ma przyjęta metoda kompresji. Kolumnowy charakter plików Parquet pozwolił na znaczne zaoszczędzenie

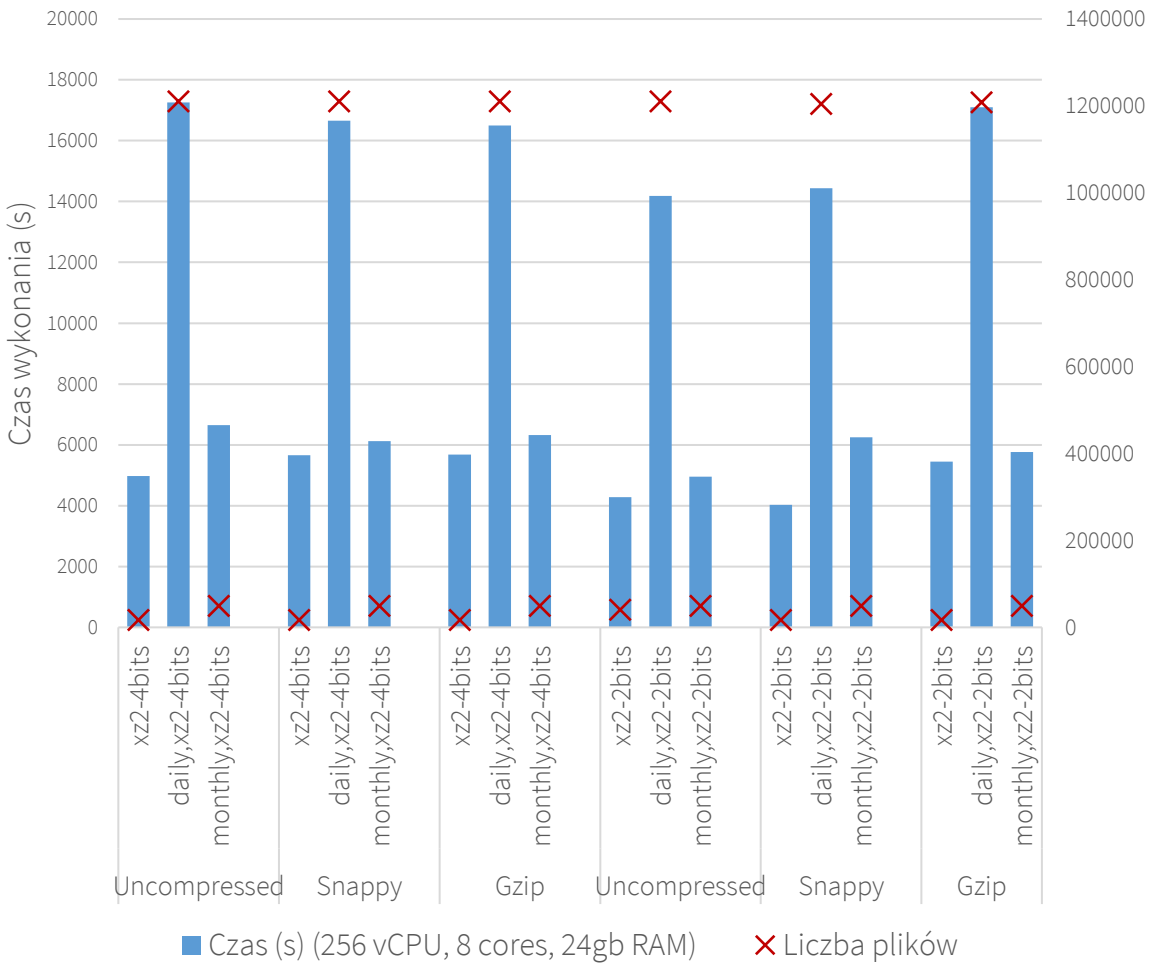
przestrzeni dyskowej w przypadku wartości wielokrotnie powtarzających się w poszczególnych kolumnach. Pliki w nieskompresowanej formie zajmowały więcej przestrzeni niż te poddane kompresji. W przypadku braku kompresji, średnia wielkość wyniosła 1169.48 GB, dla kompresji Snappy wartość ta to 843.72 GB, a dla gzip 530.25 GB. Struktura zapisu nie miała dużego wpływu na wielkość danych. W przypadku indeksowania przestrzennego na dwóch bitach, średnia wielkość wyniosła 845.43 GB, a na czterech bitach 850.20 GB. W przypadku indeksowania czasowego różnica była większa. Dane zapisane z indeksem miesięcznym zajęły średnio 824.24 GB, z dziennym 867.57 GB, a bez indeksu 851.65 GB.



Rysunek 69. Rozmiar próbki danych (w GB) na HDFS (bez replikacji) w zależności od parametrów zapisu, w porównaniu do oryginalnego rozmiaru próbki w formie pliku CSV (opracowanie własne)

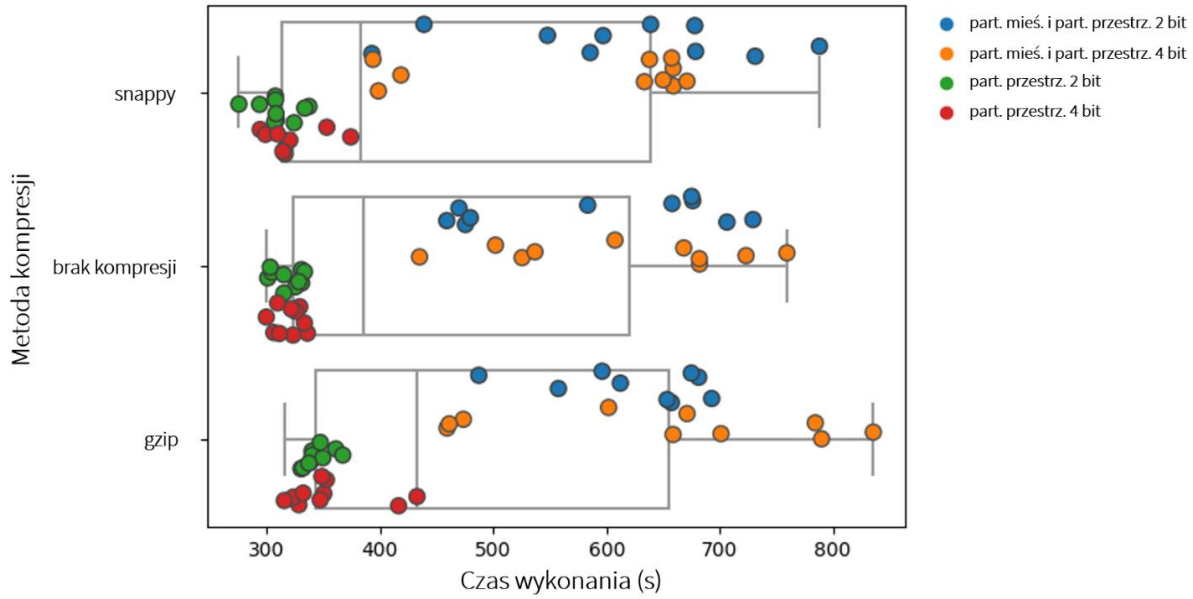
Niewielki wpływ zastosowanego indeksowania przestrzennego na wielkość zbioru jest związany z faktem, że indeks 2. oraz 4. bitowy nie powoduje podziałów danych. Indeksowanie czasowe może mieć większy wpływ w analizowanym przypadku, gdyż mocniej wpływa ono na podział plików wewnątrz struktury GeoMesa-FS co przekłada się na zróżnicowanie danych wewnątrz, a w efekcie różny efekt optymalizacji związanych z kolumnową strukturą pliku oraz zastosowaną kompresją.

Wyniki dotyczące czasu potrzebnego na zapis badanego fragmentu danych na HDFS zostały przedstawione na rysunku 70. Najmocniej zwraca uwagę fakt, iż czas potrzebny do zapisu danych z partycjonowaniem dziennym był wyraźnie dłuższy niezależnie od innych zmiennych w badaniu. Wyniósł on średnio 4 godziny 27 minut, co w porównaniu do danych zapisanych z partycjonowaniem miesięcznym (średnio 1 godzinę 40 minut) i bez partycjonowania opartego o czas (1 godzinę 23 minuty) jest wynikiem znacznie odstającym. Powodem takiego stanu rzeczy jest bardzo duża liczba plików koniecznych do utworzenia w przypadku partycjonowania dziennego. W przypadku partycjonowania dziennego średnia liczba plików wynosi 1 208 835. Dla partycjonowania miesięcznego jest to 49 156, a w przypadku braku partycjonowania liczba ta wynosi 20 323. Operowanie tak dużą liczbą małych plików na HDFS nie jest podejściem optymalnym ze względu na narzut związany z indeksowaniem ścieżek do poszczególnych plików i koniecznością odczytu wielu bardzo małych zbiorów oddzielnie.



Rysunek 70. Czas wykonania operacji zapisu oraz liczba plików w wyjściowym zbiorze danych w zależności od sposobu kompresji i zastosowanego indeksu przestrzennego (opracowanie własne)

Wnioski dotyczące wpływu liczby plików w strukturze GeoMesa-FS na wydajność powtarzają się na etapie próby odczytu całości zbioru (rysunek 71). Czas odczytu jest zdecydowanie dłuższy dla zbiorów o większej liczbie plików (z partycjonowaniem miesięcznym). W przypadku partycjonowania dziennego nie był możliwy odczyt danych w akceptowanym czasie (czas odczytu przekroczył 10 godzin oczekiwania na zakończenie operacji). Zauważyć można również dłuższy czas odczytu danych w przypadku zastosowania kompresji gzip.



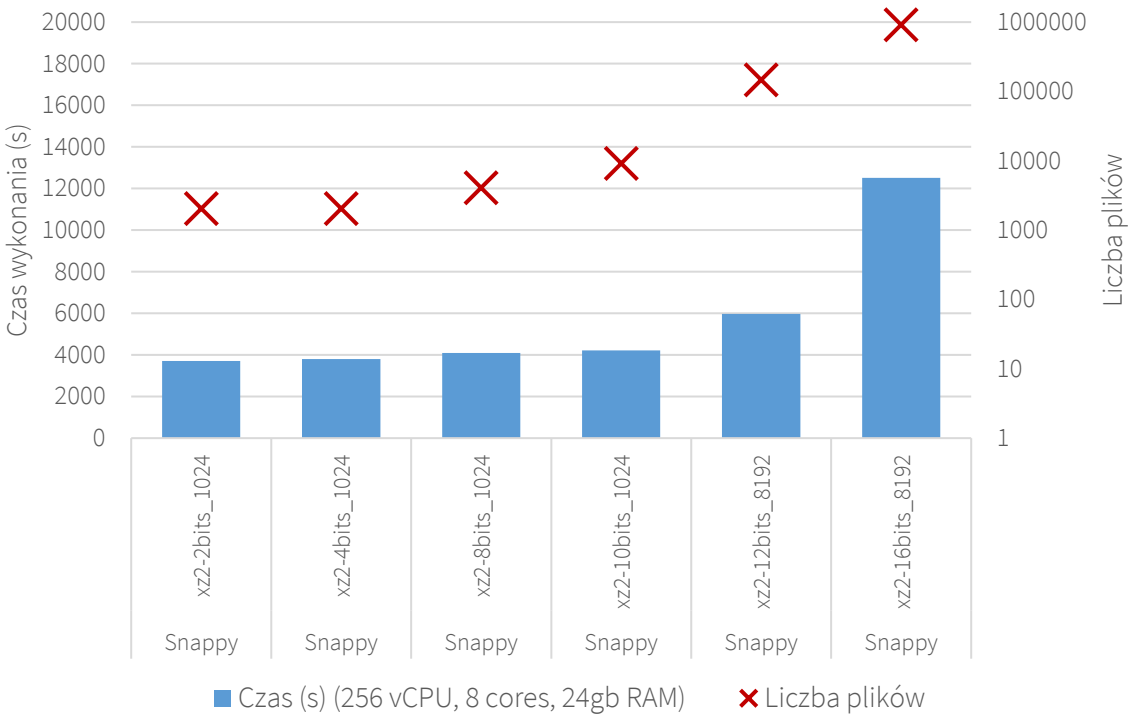
Rysunek 71. Czas wykonania operacji odczytu w zależności od sposobu kompresji i metody partycjonowania plików (opracowanie własne)

Na podstawie zestawienia statystycznego czasów odczytu danych (tabela 27) zauważyć można, że zastosowanie kompresji Snappy pozwoliło nie tylko na ograniczenie miejsca zajmowanego na dysku, ale też na wykonanie odczytu danych najszybciej. W tym przypadku czas dekompresji danych może być niższy niż czas potrzebny na pełny odczyt danych nieskompresowanych z HDFS. Czynnikiem najmocniej wpływającym na czas odczytu jest jednak zastosowanie podziału czasowego i powiązane z nim znaczne zwiększenie liczby plików.

Tabela 27. Podsumowanie statystyczne procesu odczytu fragmentu danych o średnich prędkościach pojazdów na odcinkach dróg z rozbiciem na poszczególne parametry zapisu (opracowanie własne)

Analizowany czynnik		Średnia (s)	Odchylenie standardowe (s)	Minimum (s)	Maksimum (s)
Kompresja	**Brak kompresji**	460.27	161.56	299.38	759.45
	Snappy	**454.13**	164.22	**275.10**	788.21
	Gzip	488.12	164.29	315.27	835.79
Podział przestrzenny	**2 bity**	**463.19**	159.74	**275.10**	788.21
	4 bity	471.41	166.69	293.9	835.79
Podział czasowy	**Brak**	**327.89**	26.41	**275.10**	432.53
	Miesięczny	609.14	111.97	392.62	835.79

Na podstawie pierwszej iteracji badań tego scenariusza przeprowadzono eksperyment uzupełniający mający na celu przetestowanie czasu zapisu i odczytu zbiorów przy minimalizacji liczby partycji oraz zastosowaniu większych indeksów przestrzennych (8, 10, 12 i 16 bitowego). Najmniejszą liczbę partycji, która pozwoliła poprawnie zapisać zbiór było 1024 dla partycjonowania 2 do 10 bit. W przypadku indeksów o wielkości 12 i 16 bit konieczne było zastosowanie 8192 partycji, gdyż mniejsze wartości prowadziły do błędów w zapisie danych spowodowane przepełnianiem się pamięci bądź blokowaniem obliczeń. Zastosowany indeks, ani partycjonowanie nie miało znacznego wpływu na wielkość wyjściowego zbioru i oscylowało pomiędzy 810 a 849 gigabajtów. Samo wykonanie operacji zapisu (rysunek 72) było ściśle uzależnione od wielkości indeksu (i idącą wraz z tym liczbą plików zapisywanych na HDFS). W przypadku indeksu 4. bitowego liczba plików wyniosła 2058, indeks 8. bitowy wymagał 4098 plików, 10. bitowy 9218, 12. bitowy 147 683, a 16. bitowy 917 258. Znacznie większa liczba plików w przypadku indeksów 12 oraz 16 bitowego związana jest z koniecznością zastosowania 8192 partycji zamiast 1024 zastosowanych w przypadku innych indeksów.



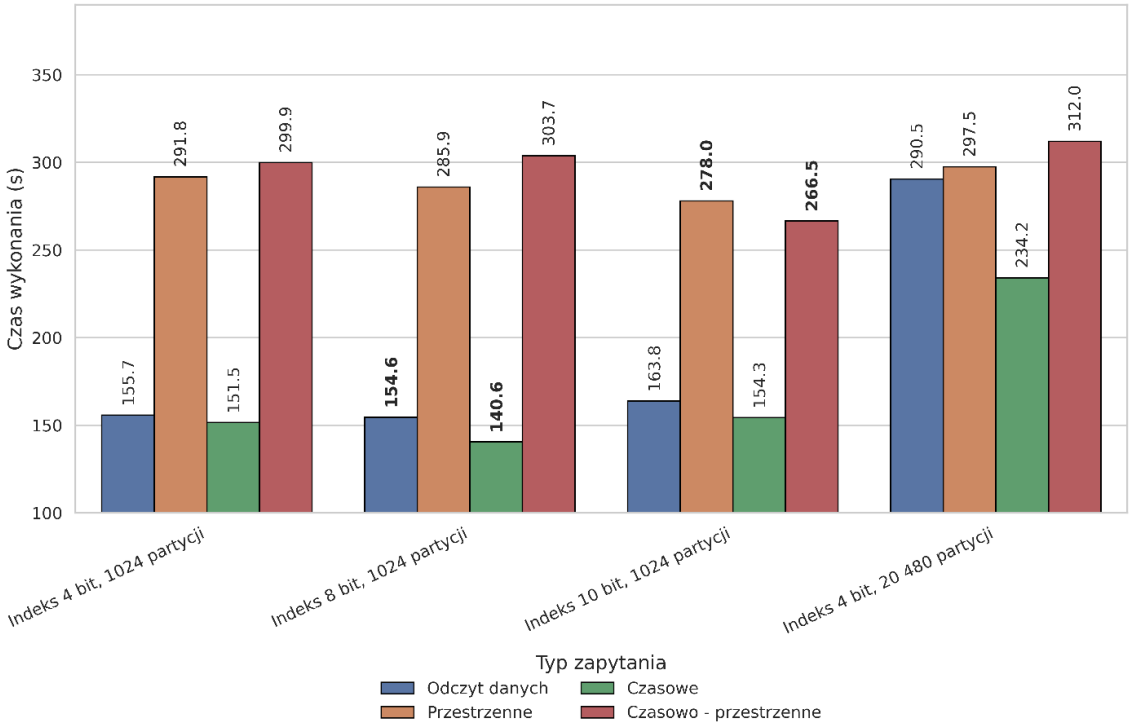
Rysunek 72. Czas wykonania operacji zapisu w zależności od wielkości indeksu przestrzennego oraz liczby plików w GeoMesa-FS (opracowanie własne)

W przypadku danych o indeksowaniu 12 oraz 16 bit nie było możliwe wykonanie operacji odczytu danych ze względu na objawy identyczne, jak w przypadku danych o dziennym indeksowaniu czasowym – zbyt duża liczba partycji powodowała brak możliwości zakończenia zadania w akceptowalnym czasie. Uśrednione czasy pomiarów z uwzględnieniem pozostałych metod indeksowania zamieszczono w tabeli 28. Na tym etapie badań nie badano już dodatkowo zastosowania indeksu czasowego ze względu na bardzo duży wzrost czasu obliczeń w związku z jego zastosowaniem w pierwszej iteracji eksperymentu.

Tabela 28. Czas wykonania operacji odczytu danych w zależności od zastosowanego indeksowania przestrzennego oraz liczby partycji wykorzystanych do zapisu danych w formacie GeoMesa-FS (opracowanie własne)

Indeks przestrzenny	Indeks czasowy	Liczba partycji	Czas wykonania [s]
4 bit	Brak	1024	**155.74**
4 bit		20 480	290.52
8 bit		1024	**154.58**
10 bit		1024	163.82

Zastosowanie zmniejszonej liczby partycji spowodowało znaczne skrócenie czasu odczytu zbioru, jednak samo indeksowanie nie wpłynęło znacznie na ten czas. W dalszej kolejności przeprowadzono analizę czasu wykonania zapytań czasowych, przestrzennych i czasowo-przestrzennych z użyciem analizowanych zbiorów danych. Średni czas wykonania tych zapytań przedstawiono poniżej (rysunek 73).



Rysunek 73. Średni czas wykonania poszczególnych typów zapytań w zależności od parametrów zapisu (opracowanie własne)

Zapytania zawróciły następujące liczby rekordów:

- Odczyt danych – 37.9 mld rekordów
- Zapytanie czasowe – 14.8 mld rekordów
- Zapytanie czasowo - przestrzenne – 287 mln rekordów
- Zapytanie przestrzenne – 716 mln rekordów

W przypadku zastosowania większych indeksów przy zachowaniu możliwie małej liczby partycji zauważyć można poprawę średniego czasu zapytań przestrzennych i czasowo – przestrzennych. Świadczy to o prawidłowym działaniu mechanizmu filtrowania na poziomie odczytu danych z dysku. Nie są to jednak bardzo duże różnice: w granicach 11% przyspieszenia względem indeksu 4 bitowego, który jest zbyt mały, żeby istotnie zmniejszyć rozmiar danych do odczytu.

Czasy wykonania zapytania czasowego były zbliżone dla wszystkich zbiorów z ograniczoną liczbą partycji. Czas wykonania zapytania czasowego krótszy niż odczyt pełnego zbioru danych wynika z zastosowania filtrowania *push-down* bezpośrednio na poziomie plików Parquet, co powoduje ograniczenie odczytu partycji nie zawierających danych z zadanego zakresu. Fragment planu fizycznego wykonania zapytania czasowego wskazuje na zastosowanie poniższej techniki filtrowania:

*…PushedFilters: [*Or(And(GreaterThanOrEqual(dtg,2019-01-01 00:00:00.0),LessThan(dtg,2019-02-01 00:00:00.0)),And(G..., ReadSchema: struct<__fid__:string,segment_id:int,dtg:timestamp,avg_speed:int,method:int,road_class:int,length...*

Warto zauważyć, że nie każda forma zapytania prowadzi do wygenerowania planu fizycznego umożliwiającego zastosowanie mechanizmu filtrowania typu *push-down*. Przykładowo, zastosowanie zapytania czasowego w postaci *WHERE month(dtg) = 1*, mimo swojej prostoty składniowej, nie skutkowało aktywacją tego mechanizmu, lecz wymuszało pełny odczyt danych z dysku. Wynika to z faktu, że filtrowanie typu *push-down* może zostać zastosowane wyłącznie w przypadku filtrów, w których nie zachodzi transformacja ani zmiana typu danych, podczas gdy użycie funkcji *month()* w warunku filtrowania wprowadza taką transformację (Konieczny, 2022).

Zbiór zapisany z domyślną liczbą partycji wymagał najwięcej czasu na wykonanie wszystkich zapytań. Należy jednak zauważyć, że różnice są zdecydowanie mniejsze w przypadku zapytań przestrzennych, niż w przypadku odczytu danych oraz zapytania czasowego. Związane jest to za faktem, że odczyt danych i zapytanie czasowe zwracają znacznie większą liczbę rekordów, a więc konieczne jest pełne odczytanie wielu partycji, co potęguje negatywny wpływ ich liczby.

Realizacja scenariusza B1 udowodniła możliwość wydajnego przechowywania i przetwarzania danych przestrzennych z wykorzystaniem środowiska SBD CENAGIS. Biorąc pod uwagę wielkość badanego zbioru danych (ponad 1TB), czasy odczytu na poziome 2-3 minut oraz prostych zapytań przestrzennych w granicach 4-5 minut można uznać za satysfakcjonujące z punktu widzenia użytkownika.

Zmiany mechanizmów indeksowania przestrzennego i czasowego danych nie przyniosły oczekiwanych wzrostów wydajności. Głównym problemem oraz ograniczeniem wydajności okazała się być liczba partycji zapisu przekładająca się bezpośrednio na liczbę plików na dysku. W przypadku większych indeksów przestrzennych (12, 16 bit) oraz czasowych (indeksowanie dzienne) zupełnie uniemożliwiło to pracę z danymi, a także ograniczyło korzyści z zastosowania indeksowania. Indeksy 8 i 10 bitowe (największe możliwe do zastosowania) nie pokrywają terenu Polski na tyle gęstą siatką, aby możliwe było uzyskanie realnych korzyści, a indeksy o wyższej rozdzielczości nie były możliwe do zastosowania. Ostatecznie więc odpowiednia optymalizacja liczby partycji w zależności od specyfiki danych jest kluczowa dla ogólnej optymalizacji obliczeń SBD, a dalsze badania nad optymalizacją indeksowania przestrzennego dla terenu Polski z użyciem metod SBD w CENAGIS przeprowadzono w ramach kolejnego scenariusza.

10.2. Scenariusz B2 – złączenie przestrzenne danych FCD

Drugi scenariusz zakłada badanie wydajności operacji złączenia przestrzennego danych punktowych (lokalizacje pojazdów) ze zróżnicowanymi pod względem liczby, stopnia skomplikowania i wielkości obszarami podziału terytorialnego kraju. Operacja złączenia przestrzennego jest jedną z podstawowych operacji na danych przestrzennych i stanowi ona podstawę wielu zaawansowanych analiz, stąd istotne jest jej przetestowanie w badanym środowisku. Złączenie przestrzenne pozwala również na analizę danych wejściowych w formie zagregowanej (np. zliczenia w gminach), co umożliwia interpretacje dużych zbiorów danych przestrzennych, których wyświetlenie w innej formie będzie niemożliwe albo skrajnie nieczytelne.

Badaniu podlegał cały zbiór danych punktów pomiarowych oraz jego podzbiory (od $1*10^{-1}$ do $1*10^{-7}$ wielkości całego zbioru) w celu zbadania różnic w wydajności poszczególnych metod w zależności od wielkości zbioru.

Do analizy włączono następujące dane obszarowe (wszystkie uprzednio ograniczone do obszaru Polski):

- Obszary gmin wg. Państwowego Rejestru Granic (PRG)
- Statystyczna siatka kilometrowa zgodna z podziałem Głównego Urzędu Statystycznego (GUS)
- Heksagony H3[74]
    - Rozdzielczość 8 (ok. 0.73 km2)
    - Rozdzielczość 9 (ok. 0.10 km2)
- Rejony statystyczne GUS
- Obwody spisowe GUS

W tabeli 29 przedstawiono parametry analizowanych zbiorów danych.

Tabela 29. Zestawienie statystyk zbiorów z poligonami podziału terytorialnego, wykorzystanych do badań wydajności złączeń przestrzennych w scenariuszu B2 (opracowanie własne)

Zbiór	L. obiektów	L. wierzchołków	Śr. l. wierzchołków na obiekt	Śr. gęstość wierzchołków na km²	Śr. powierzchnia obiektu [ha]
Gminy	2477	5 224 701	2108	24	12 161.34
Siatka kilometrowa	404 064	2 114 191	5	5	100.18
Heksagony H3 – 8	596 553	4 175 871	7	10	67.58
Heksagony H3 – 9	3 629 760	25 408 320	7	73	9.65

[74] Hierarchiczny system reprezentacji powierzchni świata w formie heksagonalnej zaprojektowany przez firmę Uber. Składa się on z 16 poziomów rozdzielczości (od 0 do 15), które połączone są ze sobą hierarchią możliwą do odtworzenie poprzez indeks przestrzenny każdego z heksagonów lub pentagonów (Brodsky, 2018).

Zbiór	L. obiektów	L. wierzchołków	Śr. l. wierzchołków na obiekt	Śr. gęstość wierzchołków na km²	Śr. powierzchnia obiektu [ha]
Rejony statystyczne	69 366	42 371 272	611	822	1689.18
Obwody spisowe	256 895	44 098 287	171	1022	266.68

Pierwszym etapem badań opartych na tym scenariuszu było wykonanie złączenia przestrzennego z wykorzystaniem podzbiorów danych o lokalizacjach GNSS oraz kilku metod złączenia przestrzennego i ich optymalizacji. Na tym etapie wykorzystano dwa zbiory obszarowe: obszary gmin oraz siatkę kilometrową.

Na tym etapie wykorzystano następujące parametry klastra: 128 x (4 vCPU, 16 GB RAM) – łącznie 512 vCPU.

Przetestowano pięć podejść do złączenia przestrzennego, które opisano poniżej.

Pierwsza metoda wykorzystuje operator *st_contains* dostępny w ramach biblioteki GeoMesa. Przykładowe zapytanie realizujące złączenie przestrzenne z użyciem tego operatora wygląda następująco:

left_df_joined = left_df.join (right_df, F.expr("st_contains(geom_left,geom_right)"),how='left')

gdzie *left_df* i *right_df* to obiekty typu Spark DataFrame zawierające odpowiednie kolumny geometryczne *geom_left* i *geom_right*. Poza optymalizacją odczytu danych z dysku w przypadku zastosowania partycjonowania przestrzennego danych oprogramowania Geomesa nie optymalizuje w dodatkowy sposób tej operacji.

Kolejna metoda stanowi rozszerzenie metody GeoMesa o ograniczenie przestrzenne danych po lewej stronie złączenia do prostokąta ograniczającego obiekt geometryczny po prawej stronie złączenia. Operacja ta pozwala na uniknięcie sprawdzania zawierania się tych obiektów, które

znajdują się poza prostokątem ograniczającym. W przypadku bardzo dużego zbioru danych po lewej stronie złączenia, takie dodatkowe ograniczenie może prowadzić do znacznej optymalizacji procesu. W tym przypadku przykładowe zapytanie będzie miało następującą postać (przy założeniu, że po lewej stronie złączenia znajdują się obiekty punktowe):

left_df_joined = left_df.join (right_df,F.expr("px > minx and px < maxx and py > miny and py < maxy and st_contains(geom_left,geom_right)"),how='left')

gdzie *px* i *py* to lokalizacja punktu, a *minx* i *miny* oraz *maxx* i *maxy* to współrzędne prostokąta ograniczającego poszczególne rekordy w kolumnie *geom_right*.

Ograniczeniem przedstawionych powyżej metod jest fakt, iż powodują one domyślne, automatyczne przesłanie mniejszego z łączonych zbiorów w trybie *broadcast*. Oznacza to, że jest on wysyłany w całości do każdej maszyny *worker* w klastrze obliczeniowym Spark. W przypadku mniejszych zbiorów danych podejście to spowoduje przyspieszenie obliczeń, jednak dla dużych zbiorów może to skutkować niepowodzeniem, ze względu na wysycenie limitów pamięci bądź ograniczeń czasowych pojedynczej operacji. Trzecia proponowana metoda złączenia przestrzennego opiera się o te same założenia, co metoda poprzednia, ale nie wykorzystuje mechanizmu *broadcast*.

Możliwe jest także wykonanie złączenia przestrzennego bez wykorzystania mechanizmów oferowanych przez biblioteką GeoMesa. Kolejna metoda wykorzystuje podejście oparte o pandas UDF. Podejście to pozwala na manualne zaimplementowanie logiki w języku Python, która potem może być wykorzystana do przeprowadzenia obliczeń rozproszonych na poszczególnych partiach danych. W tym podejściu, najpierw mniejszy zbiór kopiowany jest na maszynę *driver*, a następnie wysyłany do poszczególnych maszyn *worker*, gdzie trafiają też partie danych z większego zbioru. Tam realizowana jest funkcja *sjoin* biblioteki GeoPandas, a wynik w postaci listy identyfikatorów rekordów mniejszego zbioru jest zwracany do obiektu Spark DataFrame przechowującego dane FCD. Następnie, z użyciem tak pozyskanych identyfikatorów dokonywane jest klasyczne złączenie lewostronne, gdzie na podstawie identyfikatora dołączana jest odpowiednia geometria.

Pomimo, iż z wykorzystaniem metody opartej o pandas UDF możliwe jest przekazanie do jednostek *worker* większego zbioru niż z użyciem mechanizmu broadcast, to nadal jest ona

narażona na niepowodzenie, gdy zbiór danych znajdujący się po prawej stronie złączenia przestrzennego będzie zbyt duży. Ostatnia metoda wykorzystuje mechanizm przedstawiony powyżej, lecz rozbija zbiór znajdujący się z prawej strony złączenia na mniejsze, a operacja złączenia realizowana jest iteracyjnie na mniejszych zbiorach, co umożliwia wykorzystanie mechanizmu UDF w scenariuszu, gdy obie strony złączenia przestrzennego zawierają dużą liczbę rekordów. Wariant ten zastosowano tylko w przypadku, gdy metoda bez dzielenia zbioru nie była możliwa do wykonania ze względu na błędy przepełnienia pamięci, zbyt długiego oczekiwania na wynik i inne błędy wynikające z wysycenia zasobów obliczeniowych.

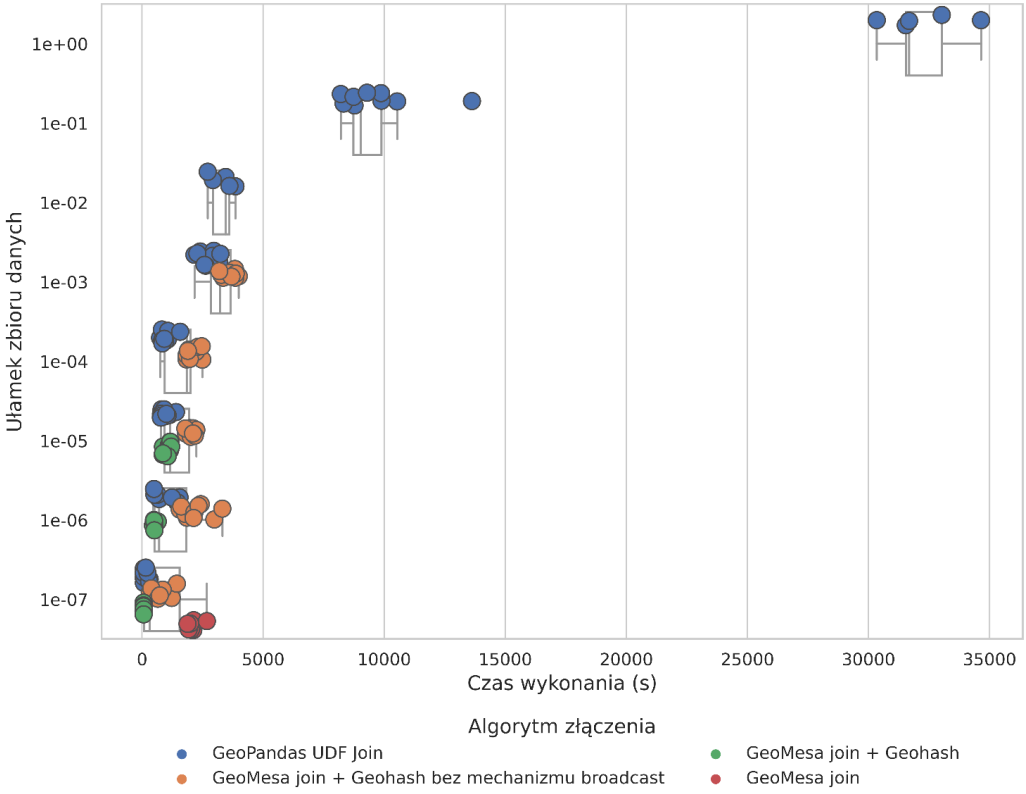
W pierwszym etapie badań wg scenariusza B2 było wykonanie złączenia przestrzennego różniących się wielkością, losowych podzbiorów danych pomiarowych FCD z warstwą gmin oraz siatką kilometrową. Złączenie wykonano z użyciem czterech różnych podejść z użyciem dostępnych w infrastrukturze CENAGIS mechanizmów SBD. W każdym z przeprowadzonych eksperymentów wynikiem działania była suma pomiarów z danej wersji zbioru w rozbiciu na poszczególne jednostki przestrzenne (gminy bądź oczka statystycznej siatki kilometrowej). Podobnie, jak w przypadku scenariusza B1 (rozdział 10.1), nie było możliwe wydajne wykorzystanie systemu partycjonowania GeoMesa-FS. W przypadku zbioru danych badanego w scenariuszu B2 możliwe było zastosowanie jedynie partycjonowania 4. bitowego przed pojawieniem się problemu ze zbyt dużą liczbą partycji. W związku z tym w tym przypadku nie jest wykorzystywana optymalizacja obliczeń na etapie odczytu danych. W poniższej tabeli zaprezentowano wielkości poszczególnych podzbiorów względem zbioru wejściowego.

Tabela 30. Wielkość podzbiorów danych utworzonych na potrzeby badań w scenariuszu B2 (opracowanie własne)

Część zbioru	Całość	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
Rozmiar na dysku	3700 GB	479 GB	66 GB	10,2 GB	1,8 GB	0,44 GB	0,22 GB	0,04 GB
Liczba rekordów	~96 mld	~9,6 mld	~960 mln	~96 mln	~9,6 mln	~960 tys.	~96 tys.	~10 tys.

W przypadku gmin (rysunek 74), wykorzystanie analizowanych metod pozwoliło na prawidłowe wykonanie analizy dla wszystkich podzbiorów, a także dla całego zbioru danych. Standardowe złączenie oferowane przez pakiet GeoMesa zadziałało poprawnie jedynie dla najmniejszego ze

zbiorów. W innych przypadkach jego zastosowanie kończyło się błędami związanymi z wysyceniem zasobów komputerowych. Podejście rozszerzone o mechanizm wstępnej filtracji zakończyło się sukcesem w przypadku zbiorów mniejszych niż 10^{-4} całości. Przy większych zbiorach mechanizm *broadcast* powodował wysycenie zasobów. Pominięcie zastosowania tego mechanizmu umożliwiło wykonanie złączenia dla zbiorów stanowiących 10^{-3} całości i mniejszych. Metoda ta byłą jednak już wyraźnie wolniejsza od innych stosowanych podejść. Rozwiązaniem najwydajniejszym przy większych zbiorach danych, a także jedynym, które pozwoliło na wykonanie złączenia na pełnym zbiorze było podejście oparte o funkcję UDF ze złączeniem GeoPandas. Od wielkości 1/100 całego zbioru oraz większych była to jedyna działająca poprawnie metoda zdolna do wykonania założonego scenariusza.

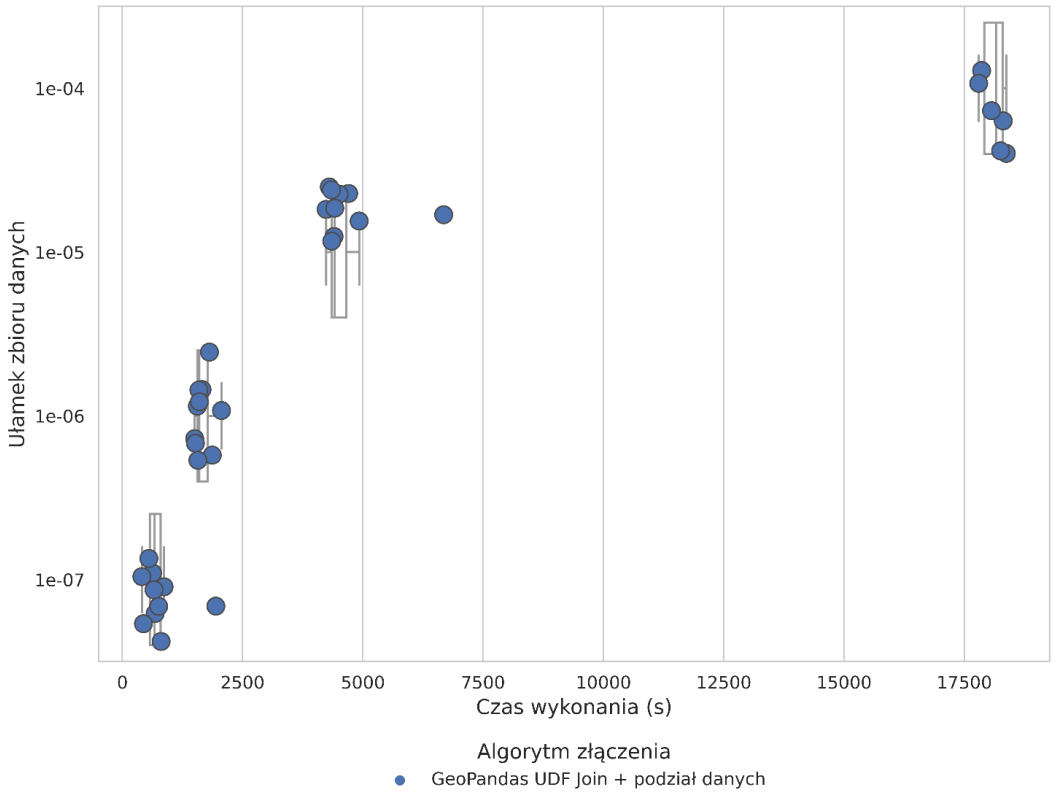


Rysunek 74. Czas wykonania operacji złączenia przestrzennego dla obszarów gmin w zależności od zastosowanej metody złączenia i wielkości zbioru danych wejściowych (opracowanie własne)

Dla stosunkowo małych zbiorów (do ok. 100 tysięcy punktów) metoda oparta o złączenia GeoMesa z filtrowaniem wykazywała najwyższą wydajność. Eliminacja mechanizmu *broadcast* pozwoliła na włączenie do analizy większej liczby punktów, ale skutkowało to znacznym spadkiem wydajności (dla zbioru 10^{-6} był to ponad 4-krotny spadek wydajności). Metoda oparta o funkcję UDF była mniej wydajna przy mniejszych zbiorach, ale w innych przypadkach

cechowała się wyższą wydajnością i stabilnością działania. Wykonanie pełnego złączenia na wszystkich punktach z użyciem tej metody zajęło blisko 9 godzin.

Złączenie punktów z siatką kilometrową (rysunek 75) było możliwe jedynie z wykorzystaniem metody opartej o funkcję UDF z dodatkowym podziałem zbioru stanowiącego prawą cześć złączenia. W przypadku innych metod nawet najmniejsza wersja zbioru nie pozwalała na poprawne zakończenie obliczeń. Największym zbiorem, dla którego możliwe było wykonanie całości obliczeń była zbiór stanowiący 1/10000 całości punktów pomiarowych. Na tym pułapie punktów czas obliczeń wyniósł już jednak ponad 5 godzin.



Rysunek 75. Czas wykonania operacji złączenia przestrzennego dla siatki kilometrowej w zależności od zastosowanej metody złączenia i wielkości zbioru danych wejściowych (opracowanie własne)

Możliwe byłoby więc podzielenie zbioru na mniejsze podzbiory i wykonania złączenia iteracyjnie dla całego zbioru. Zajęłoby to ponad 50 tysięcy godzin, co przekłada się na 2096 dni. Wykorzystanie badanych podejść nie pozwala więc na wykonanie tego złączenia w akceptowalnym czasie.

Pierwszy etap badań wykazał, że zastosowanie opisanych powyżej metod złączenia nie pozwala na wykonanie złączenia z użyciem pełnego zbioru danych o lokalizacjach dla wszystkich

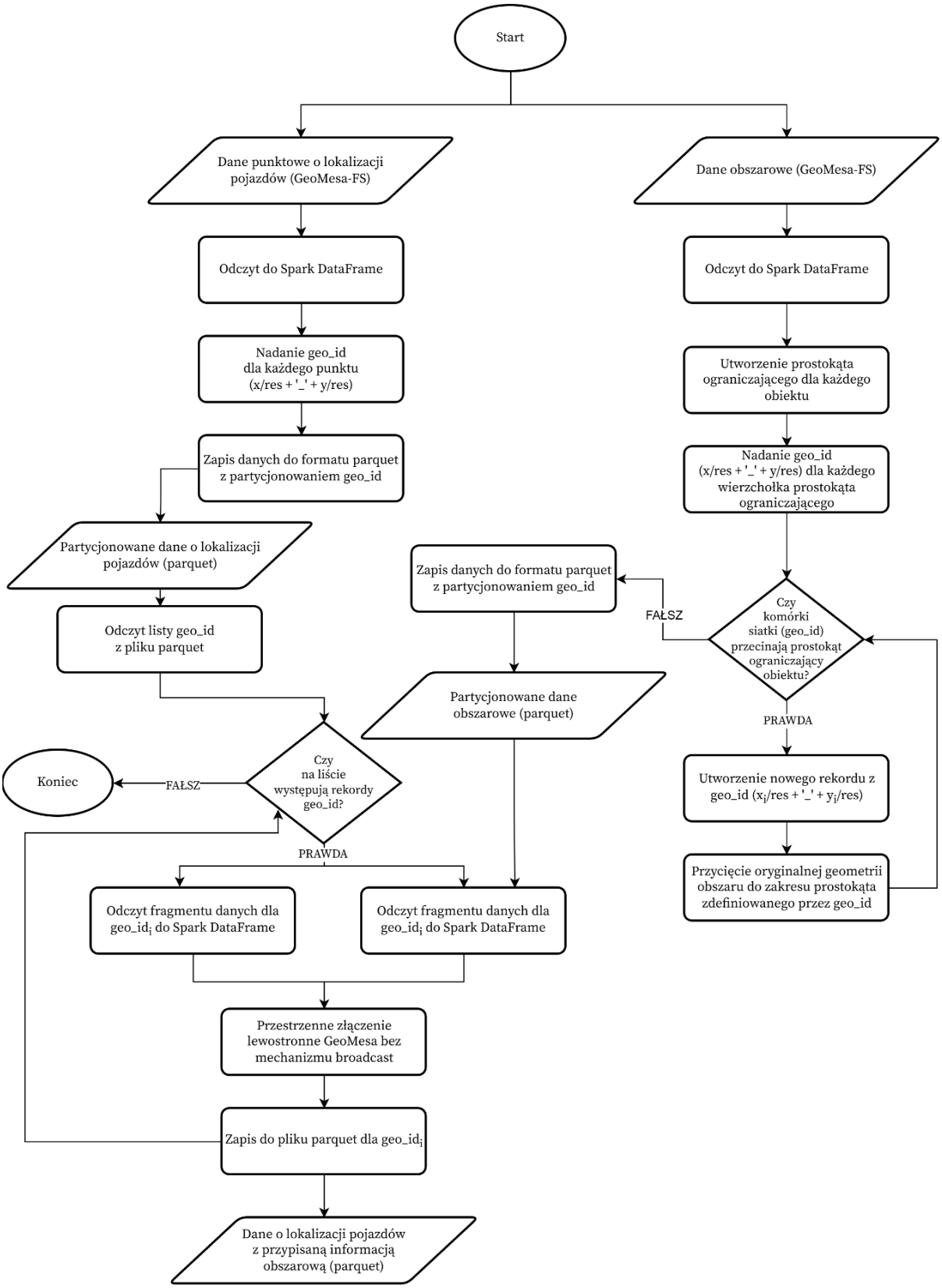
z użytych zbiorów danych poligonowych. Metoda oparta o mechanizm pandas UDF pozwoliła na wykonanie złączenia z pełnym zbiorem danych jedynie dla poligonów gmin, inne metody kończyły się niepowodzeniem z powodu wysycenia przypisanych zasobów, osiągnięcia systemowych limitów oczekiwania na odpowiedź bądź zawieszenia się przetwarzania.

W drugim etapie opracowano więc zoptymalizowaną metodę rozproszonego złączenia przestrzennego (rysunek 76), którą następnie przetestowano z użyciem wszystkich analizowanych zbiorów. Kluczowym elementem proponowanej metody jest wprowadzenie pomocniczego indeksu przestrzennego *geo_id*, który umożliwia partycjonowanie danych w oparciu o regularną siatkę przestrzenną oraz ograniczenie zakresu danych przetwarzanych w ramach poszczególnych operacji.

W pierwszym etapie dane punktowe i obszarowe są niezależnie wczytywane do struktur Spark DataFrame. Dla danych punktowych wyznaczany jest identyfikator *geo_id* na podstawie współrzędnych przestrzennych, a następnie dane zapisywane są do formatu Parquet z partycjonowaniem według tego identyfikatora. Równolegle, dla danych obszarowych wyznaczany jest prostokąt ograniczający każdego obiektu, który następnie transformowany jest na zbiór komórek siatki *geo_id*. Dla każdej komórki przecinającej prostokąt ograniczający tworzony jest odpowiedni rekord, a geometria obiektu poligonowego jest przycinana do zakresu danej komórki. Tak przygotowane dane obszarowe również zapisywane są do formatu Parquet z partycjonowaniem według kolumny *geo_id*.

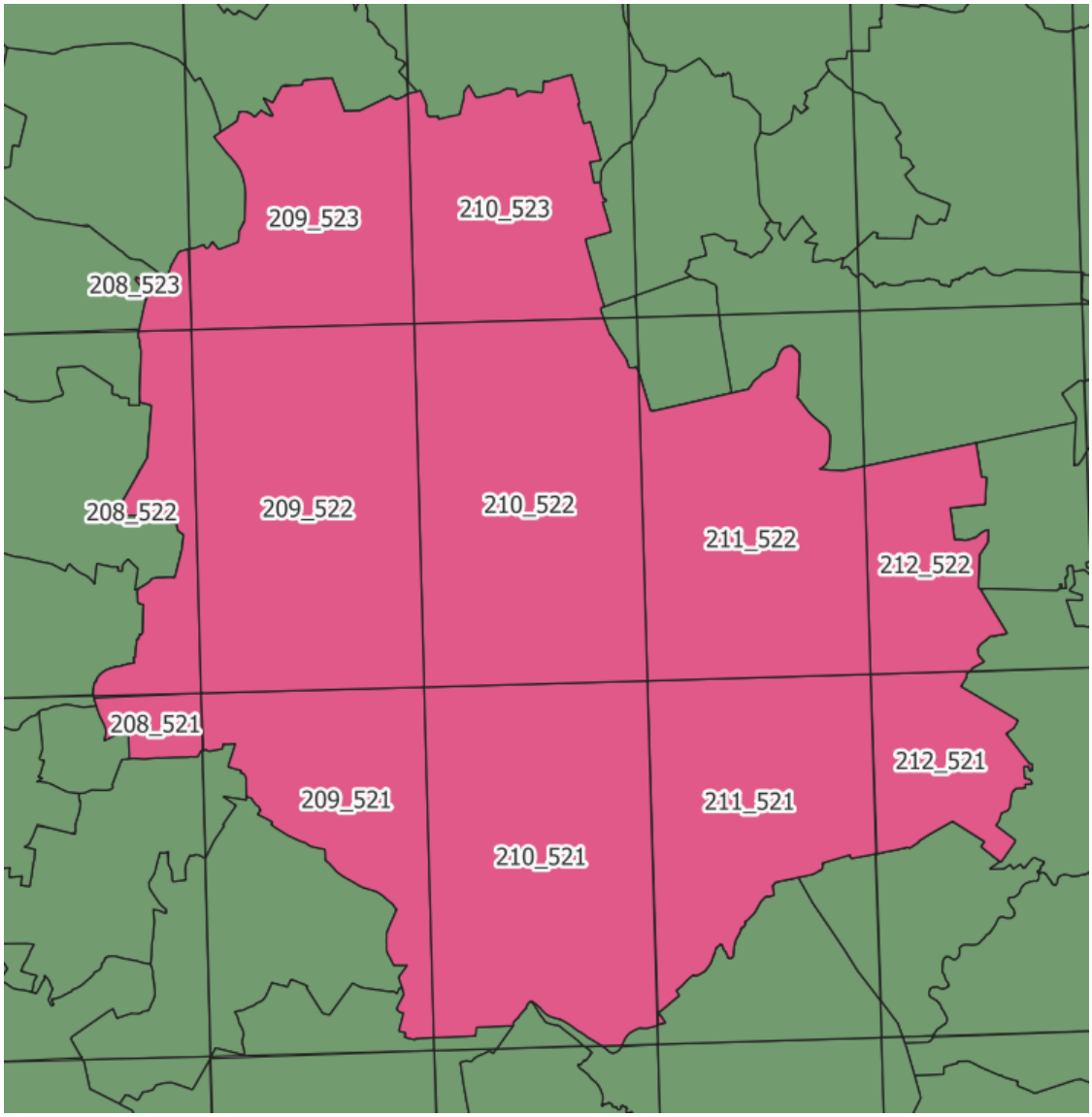
W kolejnym etapie realizowane jest iteracyjne przetwarzanie danych w obrębie kolejnych partycji *geo_id*. Dla każdej z nich odczytywane są odpowiadające fragmenty danych punktowych oraz obszarowych, po czym wykonywane jest lewostronne złączenie przestrzenne z wykorzystaniem mechanizmów GeoMesa, bez zastosowania strategii *broadcast* (podejście to opisane je szerzej w pierwszej części tego rozdziału).

Wynikiem działania algorytmu jest zbiór danych o lokalizacji pojazdów zapisany w formacie Parquet, rozszerzony o przypisaną informację obszarową. Zaproponowany proces umożliwia efektywne wykonanie złączeń przestrzennych dla dużych wolumenów danych poprzez świadome wykorzystanie partycjonowania przestrzennego i kontrolę zakresu danych przetwarzanych w każdej iteracji.



Rysunek 76. Schemat blokowy opracowanego w ramach rozprawy procesu rozproszonego złączenia przestrzennego (opracowanie własne)

Na tym etapie badań sprawdzono, czy dodatkowa optymalizacja polegająca na przycinaniu geometrii poligonowej do granic komórki indeksu *geo_id* wpływa na czas przetwarzania. Zabieg ten miał ograniczyć koszt operacji przestrzennych w przypadkach, gdy obiekt poligonowy był przypisany do danej komórki, lecz jego właściwa geometria w dużej części znajdowała się poza analizowanym zakresem. W takim układzie wykonywanie testów przestrzennych na pełnej geometrii generuje niepotrzebny narzut obliczeniowy, dlatego w badaniu porównano wariant bez przycinania z wariantem, w którym geometria jest redukowana do fragmentu zawartego w zakresie komórki (rysunek 77).



Rysunek 77. Przykład danych wejściowych po przycinaniu geometrii poligonowej do zakresu komórek indeksu przestrzennego o rozmiarze 0,1° na obszarze granic miasta Warszawy. Każdy wyróżniony fragment (kolor różowy) odpowiada odrębnej wartości *geo_id* i stanowi część oryginalnej geometrii ograniczoną do granic właściwej komórki indeksu (opracowanie własne)

Badania na tym etapie przeprowadzono w następującej konfiguracji klastra:

64 x (8 vCPU, 16 GB RAM) – łącznie 512 vCPU.

W przypadku rozproszonego złączenia przestrzennego konieczne było sprawdzenie kilku parametrów tego procesu w celu optymalizacji dalszych badań na innych zbiorach. Do głównych badanych parametrów należały:

- Sposób wykonywania obliczeń w Spark – sekwencyjny albo współbieżny. Jest to kluczowy parametr ze względu na fakt, że w podejściu rozproszonym obliczenia Spark wykonywane są na mniejszych obszarach. W przypadku podejścia sekwencyjnego obliczenia wykonywane są obszar po obszarze, co może powodować obniżenie wydajności, gdyż wszystkie dostępne moce obliczeniowe zostaną przeniesione na jeden obszar na raz. W przypadku, gdy liczba partycji na które podzielony jest zbiór na tym obszarze będzie zbyt mała (mniejsza niż maksymalna wykorzystania liczba CPU) może dojść do spowolnienia obliczeń.
- Liczba procesów w podejściu współbieżnym. Przy zastosowaniu podejścia współbieżnego kolejnym istotnym parametrem jest maksymalna liczba wywołanych na raz procesów Spark. Mniejsza liczba może tylko częściowo zniwelować problemy podejścia sekwencyjnego. Z kolei za duża może doprowadzić do niepotrzebnie dużej liczby procesów, nadmiernego obciążenia maszyny „Driver" czy powstania ograniczenia wydajności na etapie zapisu danych wyjściowych na dysk.

W badaniach przyjęto wielkość partycji przestrzennej na 0.1x0.1 stopnia. Wielkość partycji również ma wpływ na wydajność obliczeń i jest zależna od charakterystyki danych. Wstępnie analizy wskazały wartość 0.1 stopnia, jako dobrą dla badanych zbiorów, lecz jest to potencjalne pole do dalszej analizy. W przypadku wykonania złączenia należy również uwzględnić czas potrzebny na wstępne przygotowanie danych (odczyt, partycjonowanie i zapis) zarówno punktowych, jak i poligonowych. Jest to jednak czynność jednorazowa dla pojedynczego zbioru danych i po przygotowaniu mogą być one użyte wielokrotnie do różnych złączeń przestrzennych czy innych operacji. W przypadku zbioru danych FCD operacja ta zajęła średnio ok. 2 godziny 52 minuty. Dla każdego z badanych zbiorów policzono także czasy przygotowania samego zbioru do złączenia, a także czas przygotowania odpowiedniej kwerendy Spark, gdyż

przy dużej liczbie zapytań proces ten nie jest natychmiastowy. Badania rozpoczęto na zbiorze, który uniemożliwił poprawne wykonanie obliczeń uprzednio tj. na siatce kilometrowej.

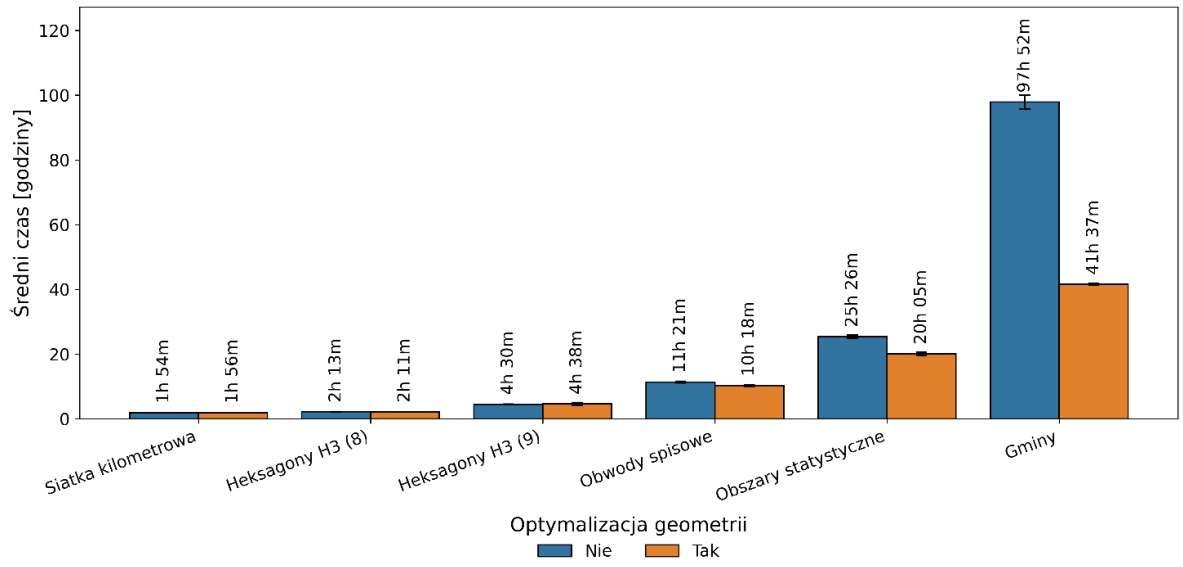
W pierwszym teście porównano podejścia oparte o współbieżne i sekwencyjne konstruowanie zapytań. W przypadku podejścia sekwencyjnego operacja zakończyła się po 78 godzinach obliczeń. W przypadku podejścia współbieżnego czas ten został skrócony do niecałych 5 godzin przy 16 procesach i do niecałych 2 godzin przy 64 procesach. Dalsze zwiększanie liczby procesów nie przyniosło przyspieszenia obliczeń. Ze względu na bardzo dużą różnicę w czasie realizacji zapytań dalsze badania oparto już wyłącznie o podejście współbieżne. Czas przygotowania danych wyniósł średnio 11 minut 46 sekund w przypadku zastosowania optymalizacji geometrii oraz 8 minut 57 sekund bez tej optymalizacji. Zauważyć można, że w przypadku siatki kilometrowej optymalizacja geometrii nie przyniosła przyspieszenia obliczeń, a wręcz je minimalnie spowolniła. Zaproponowane podejście pozwoliło na wykonanie złączenia przestrzennego na pełnym zbiorze danych FCD z siatką kilometrową, co nie było możliwe do wykonania z użyciem innych podejść.

W przypadku heksagonów H3 o rozdzielczości 8 średni czas obliczeń wyniósł 2. godziny 13 minut bez optymalizacji geometrii oraz 2. godziny 11 minut z optymalizacją. Zastosowanie jej w tym przypadku przynosi minimalne przyspieszenie obliczeń. Jest ono jednak niwelowane przez dłuższy czas przygotowania danych w przypadku optymalizacji (o 2 minuty 33 sekundy). Ponownie, podobnie, jak w przypadku siatki kilometrowej zastosowanie większej liczby procesów przełożyło się na znaczne przyspieszenie obliczeń. W przypadku heksagonów H3 o rozdzielczości 9 średni czas obliczeń wyniósł ok. dwa razy dłużej niż w przypadku heksagonów o rozdzielczości 8: 4 godziny 30 minut bez optymalizacji geometrii oraz 4 godziny 38 minut z optymalizacją. Ponownie obliczenia z użyciem 64 procesów przyniosły znaczny wzrost wydajności. W dalszych testach nie był już sprawdzany czas obliczeń dla 16 procesów ze względu na jednoznaczne wnioski dotyczące wpływu wydajności tego parametru. Za każdym razem liczba 64 procesów powodowała pełne wykorzystanie założonych mocy w rozbiciu na aktualnie trwające procesy.

Złączenie przestrzenne dla zbioru danych o obwodach spisowych GUS okazało się być znacząco bardziej wydajne w przypadku zastosowania optymalizacji geometrii. Proces bez optymalizacji zajął średnio 11 godzin 21 minut, a z optymalizacją 10 godzin 18 minut, co

przekłada się na 9.1% przyspieszenia obliczeń. Zauważyć można tutaj związek złożoności oraz średniej wielkości geometrii z zasadnością wdrożenia optymalizacji. Parametry te mają również wpływ na czas przygotowania danych, w przypadku zbioru obwodów spisowych przygotowanie zbioru do złączenia zajęło średnio 25 minut 57 sekund. W przypadku zbioru obszarów statystycznych GUS również zachodzi przyspieszenie obliczeń z wykorzystaniem optymalizacji geometrii z ok. 25 godzin 27 minut do ok. 20 godzin 5 minut (co przekłada się na 21% przyspieszenia).

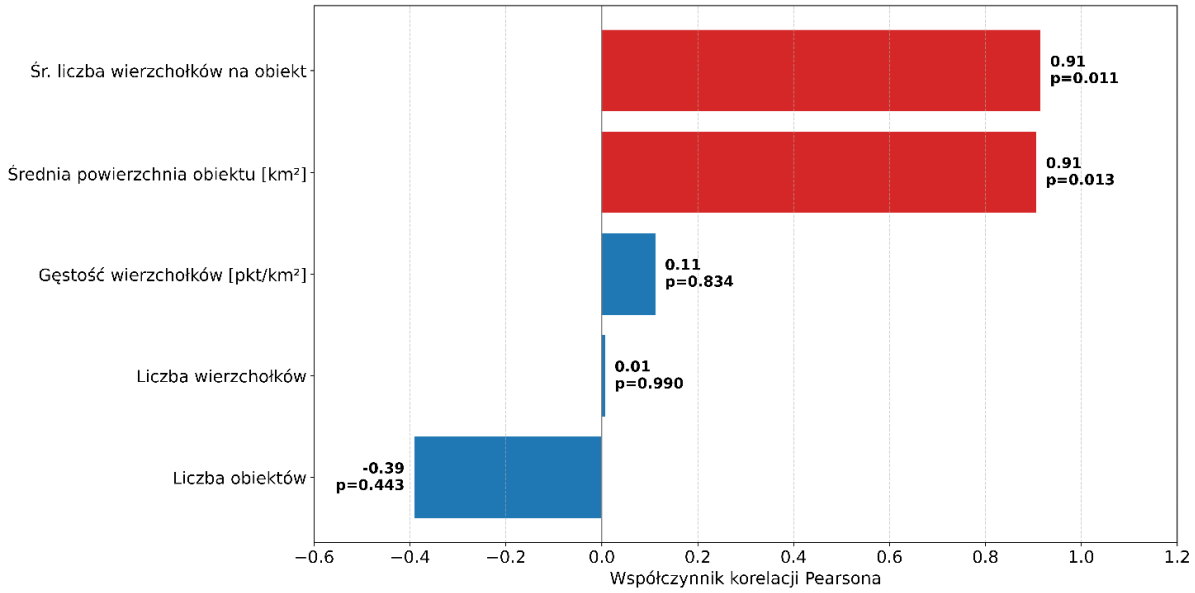
Podejście rozproszone w przypadku analizy zbioru z granicami gmin okazało się być mało wydajne. osiągając czas obliczeń znacznie gorszy niż w przypadku złączenia opartego o funkcję UDF, które było możliwe dla całego zbioru. Z zastosowaniem optymalizacji średni czas obliczeń wyniósł 41 godzin 37 minut, a bez jej zastosowania było to aż 97 godzin 52 minuty. Ta sama operacja z użyciem funkcji UDF zajęła średnio 8 godzin 57 minut. Na rysunku 78 przedstawiono podsumowanie czasu obliczeń dla poszczególnych zbiorów z uwzględnieniem optymalizacji geometrii zbiorów poligonowych.



Rysunek 78. Zestawienie średnich czasów wykonania złączenia przestrzennego z wykorzystaniem metody złączenia rozproszonego dla wszystkich badanych obszarów (opracowanie własne)

Rozproszone złączenie przestrzenne cechowało się bardzo zróżnicowanymi czasami obliczeń w zależności od charakterystyki zbioru. Wykonano analizę korelacji pomiędzy parametrami warstw poligonowych, a czasem złączenia (rysunek 79) w celu lepszego zrozumienia, które z cech zbiorów najmocniej wpłynęły na wyniki. Na podstawie przeanalizowanych zbiorów zauważyć można, że największy wpływ na czas obliczeń miała powierzchnia obiektów oraz

poziom ich złożoności (liczba wierzchołków przypadająca na obiekt). W przypadku obliczeń rozproszonych jest to zrozumiałe, gdyż duże obiekty będą wielokrotnie wczytywane i analizowane w ramach wielu partycji przestrzennych.



Rysunek 79. Analiza korelacji pomiędzy parametrami warstw poligonowych, a czasem złączenia wykonanego metodą rozproszoną (opracowanie własne)

W takich przypadkach największą korzyść przynosi zastosowanie optymalizacji polegającej na przycinaniu geometrii do granic analizowanych partycji przestrzennych – w ten sposób niwelowany jest wpływ zarówno złożoności obiektu, jak i jego wielkości, gdyż analizie podlega tylko ten fragment geometrii, który jest konieczny do sprawdzenia relacji zawierania. Wyniki opracowania w postaci wizualizacji kartograficznych (zliczenie rekordów w poszczególnych obszarach) zamieszczono w załączniku 2.

Dodatkowo sprawdzono jak badana metoda sprawdza się przy analizach z użyciem mniejszych zbiorów danych. Wykonano więc badanie wydajności metody na danych opisywanych w badaniach na danych wektorowych w drugim etapie badań. Złączenie zostało dokonane na podstawie danych FCD dla jednego dnia pomiarów w agregacji do obszarów gmin z użyciem następujących parametrów klastra: 50 x (4 vCPU, 32 GB RAM).

Uzyskane wyniki są znacząco gorsze niż podejście oparte o funkcję UDF, gdzie całość obliczeń zajęła 86 sekund. W przypadku złączenia z użyciem 64 procesów czas ten wyniósł 33 minuty 58 sekund. Dodatkowo, przygotowanie zbioru punktów FCD dla tego jednego dnia zajęło średnio 23 minuty 19 sekund.

Zaproponowana metoda pozwoliła na prawidłowe złączenie wszystkich analizowanych warstw z całym zbiorem punktowych danych FCD. Główna trudność, której pokonanie nie było możliwe z użyciem innych podejść do złączenia przestrzennego w Spark, wynikała z faktu dużego rozmiaru zbioru danych poligonowych. W przypadku mniejszych zbiorów danych po prawej stronie złączenia (np. gminy) zastosowanie metod niewymagających przebudowy zbioru danych do innej postaci na dysku przyniesie lepszy wynik. Im mniejszy zbiór tym większe możliwości przyspieszenia obliczeń np. z wykorzystaniem systemu *broadcast*. W badanym systemie nie istnieje więc jeden optymalny sposób wykonania złączenia przestrzennego na dużych zbiorach danych, a bardzo wiele zależy od specyfiki zbioru po obu stronach złączenia (w przypadku zbiorów liniowych bądź poligonowych po lewej stronie złączenia należałoby zastosować modyfikacje przedstawionych metod). Następnym wyzwaniem utrudniającym realizację scenariusza był fakt, że złączenie, w każdym przypadku dotyczyło zakresu całej Polski. Oznacza to, że nie było możliwości uniknięcia odczytu całości zbioru punktowego FCD z wykorzystaniem np. metod filtracji przestrzennej na etapie odczytu danych. Przygotowanie metod zdolnych przeprowadzić pełne złączenie na bardziej złożonych zbiorach danych było bardzo czasochłonne, a ze względu na braki w dokumentacji stosowanych rozwiązań, a także różnice w najnowszych wersjach poszczególnych pakietów oprogramowania, a wersjami stosowanymi w CENAGIS powodowały, że konieczne było stosowanie metody prób i błędów w celu osiągnięcia funkcjonalnego rozwiązania.

10.3. Scenariusz B3 – obliczenie współczynnika LOS

Trzeci scenariusz opierał się o operacje na danych o średniej prędkości na odcinkach dróg. Dla całego zbioru danych obliczono tzw. współczynnik LOS (ang. *Level of Service*). Współczynnik ten mówi o poziomie obciążenia danego odcinka ruchem ulicznym względem jego maksymalnej możliwej przepustowości. Został on zaproponowany po raz pierwszy w 1965 roku w *Highway Capacity Manual* (Transportation Research Board, 1965) jako miara w skali od A do F. W tabeli 31 przedstawiono charakterystykę ruchu dla każdego poziomu LOS.

Tabela 31. Opis poziomów Level of Service. Na podstawie *(Departament Transportu USA, 2021)*

Level of Service (LOS)	Opis
A	Swobodny przepływ, niskie natężenie ruchu i wysokie prędkości.
B	Ruch w dużej mierze swobodny, ale prędkości zaczynają być ograniczane przez warunki ruchu.
C	Przepływ stabilny, większość kierowców ma ograniczoną swobodę wyboru prędkości.
D	Przepływ zbliżający się do niestabilnego; kierowcy mają niewielką swobodę wyboru prędkości.
E	Przepływ niestabilny; mogą występować krótkie zatrzymania.
F	Wymuszony lub załamany przepływ; niedopuszczalne zatłoczenie; jazda w trybie *stop-and-go*.

Współczynnik ten pozwala określić ogólny stan obciążenia drogi i jest często wykorzystywany w procesie decyzyjnym dotyczącym inwestycji drogowych (Litman, 2003), nie tylko w odniesieniu do ruchu samochodowego, ale po odpowiednich modyfikacjach również w innych kontekstach tj. transportu publicznego (Wang, Mao, Yang, Shi i Huang, 2022) czy rowerowego (Kazemzadeh i Ronchi, 2022)

Współczynnik LOS można liczyć w odniesieniu zarówno do liczby aut, jak i do ich prędkości Generalnie współczynnik wyliczany jest według wzoru:

$$LOS = \frac{V}{C} = \frac{N_v}{N_{max}}$$

Gdzie: $\boldsymbol{V/C}$ to *volume-to-capacity ratio* (współczynnik objętości do pojemności), $\boldsymbol{N_v}$- chwilowe obciążenie ruchu, $\boldsymbol{N_{max}}$- maksymalne obciążenie ruchu.

W przypadku, gdy analiza LOS oparta jest o prędkości pojazdów, wzór przyjmuje postać:

$$LOS = \frac{V}{V_{FFS}}$$

Gdzie: V – średnia prędkość na danym odcinku, a V_{FFS} - prędkość na tym odcinku w ruchu swobodnym – bez żadnych ograniczeń związanych z ruchem.

Na podstawie otrzymanej wartości możliwe jest przypisanie oceny od A do F mówiącej o przełożeniu wartości współczynnika na odczucia uczestników ruchu drogowego. Według Highway Capacity Manual (Transportation Research Board, 1994) w przypadku wykorzystania informacji o prędkości należy przyjąć następujące, orientacyjne wartości:

Tabela 32. Poziom Level of Service w zależności aktualnej prędkości wyrażonej jako % prędkości w ruchu swobodnym na podstawie (Transportation Research Board, 1994).

Poziom LOS	A	B	C	D	E	F
% prędkości V_{FFS}	>90	>70	>50	>40	>30	<=30

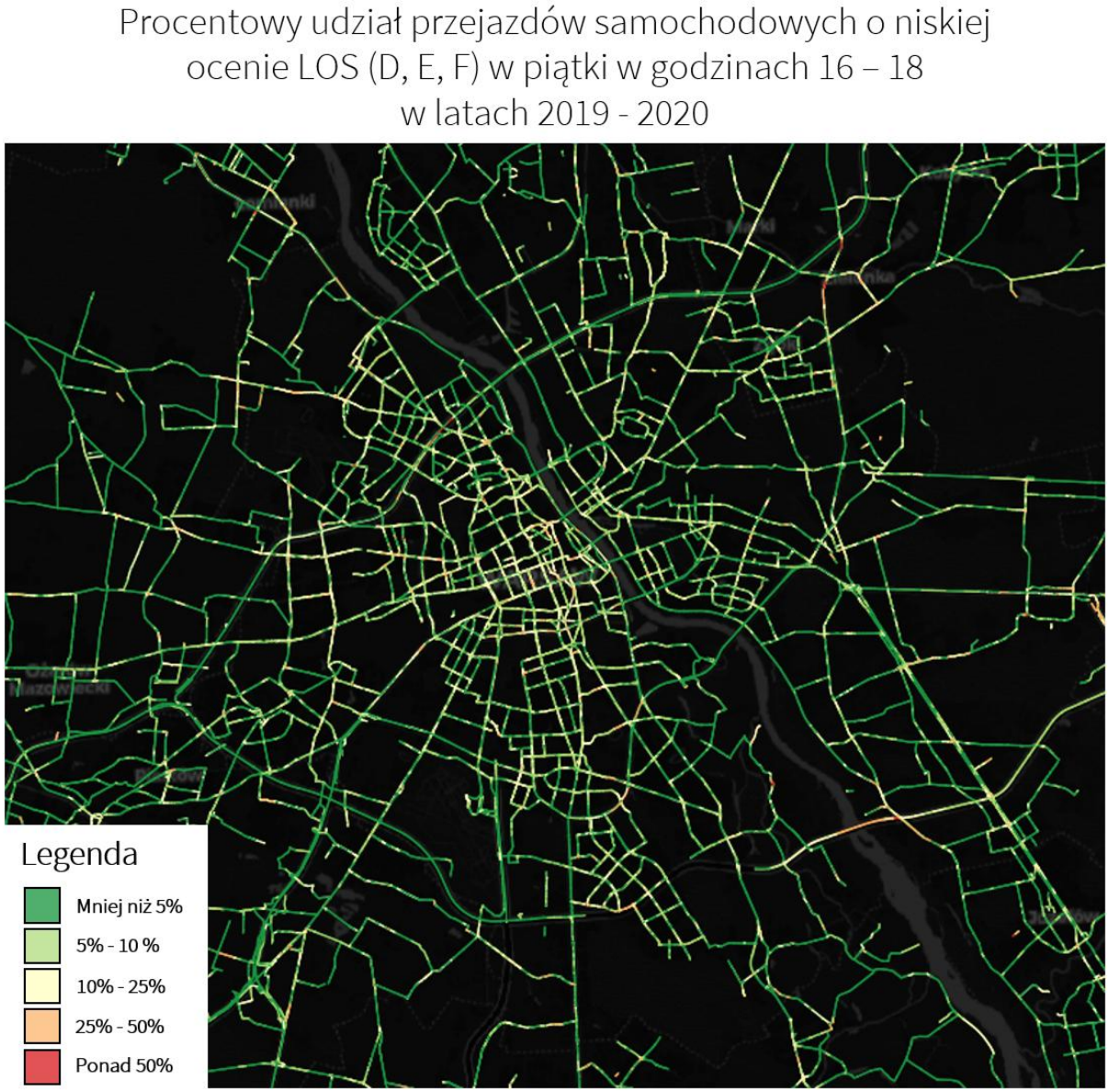
Badania wydajności w tym scenariuszu składały się z trzech etapów przetwarzania:

- Wyliczenie prędkości w ruchu swobodnym na podstawie danych z pojazdów pozyskiwanych w czasie godzin nocnych dla każdego segmentu drogi (na podstawie danych z godzin nocnych – metoda wykorzystana m.in. w (Ambros i Kyselý, 2016) oraz przez firmę Here[75])
- Obliczenie parametru LOS dla całego zbioru danych na podstawie pozostałych danych
- Filtracja danych do zadanego zakresu czasowego (ze względu na przydatność praktyczną analizy wybrano okres wzmożonego ruchu drogowego - piątki w godzinach 16-18), agregacja do postaci pojedynczego współczynnika dla każdego segmentu i zapis do formatu łatwego do wizualizacji w narzędziach GIS (ESRI SHAPEFILE). Współczynnik zawiera informację o stosunku występowania pozytywnych współczynników LOS (A,B,C) do wszystkich. Ta informacja pozwala na ocenę danego segmentu drogi

[75]https://www.here.com/docs/bundle/traffic-analytics-developer-guide/page/topics/free-flow-speed-freeflow.html (data dostępu 15.12.2025)

w odniesieniu do całego badanego czasowego zakresu analizy w formie jednej wartości liczbowej

W scenariuszu tym realizowane są zadania nieprzestrzennych złączeń danych oraz agregacji, a także zapisu do formatu łatwo interpretowalnego przez klasyczne narzędzia GIS. Na rysunku 80 pokazano fragment wizualizacji wyniku przeprowadzonej analizy. Uzyskana mapa pokazuje procentowy udział przejazdów o niskim LOS w wybrane dni i godziny tygodnia w wybranych latach. Żółta, pomarańczowa i czerwona barwa pokazują miejsca, w których najczęściej dochodzi do pogorszenia jakości ruchu samochodowego.



Rysunek 80. Wizualizacja fragmentu danych wyjściowych (Warszawa i okolice) z obliczonym współczynnikiem liczby wystąpień współczynnika LOS od D do F do wszystkich wystąpień dla rekordów zarejestrowanych w piątki w godzinach 16-18 (opracowanie własne), podkład mapowy: OpenStreetMaps

Scenariusz zrealizowano przy wykorzystaniu następujących parametrów klastra:

64 x (8 vCPU, 16 GB RAM) – łącznie 512 vCPU.

Zadanie obliczenia współczynnika LOS rozdzielono na trzy etapy:

1. Przygotowanie zbioru prędkości pojazdów w ruchu swobodnym
2. Obliczenie parametru *Level of Service* dla całego zbioru danych
3. Generowanie danych zagregowanych – charakterystyka LOS dla ruchu w piątki wieczorem

Obliczenia zostały przeprowadzone w oparciu o pełny zbiór danych o średnich prędkościach na odcinkach dróg w postaci GeoMesa-FS. Łączny rozmiar danych wejściowych wyniósł 8.9 TB, co przekłada się na 423.5 miliarda rekordów.

Realizacja etapu 1 składała się z następujących kroków:

1. Odczyt pełnego zbioru danych o prędkościach
2. Filtracja danych reprezentujących godziny rejestracji od 23 do 5
3. Grupowanie danych o wspólnym identyfikatorze segmentu i policzenie średniej wartości prędkości dla każdego z segmentów
4. Zapis wyników (identyfikator segmentu oraz średnia prędkość w ruchu swobodnym) do pliku Parquet na HDFS

Wynikowy zbiór danych zajął 24.3 MB, czas przygotowania pliku z prędkościami swobodnymi wyniósł średnio 25 min 40 sekund.

W kolejnym etapie przypisano informacje o współczynniku LOS do każdego rekordu w danych wejściowych:

1. Odczyt pełnego zbioru danych o prędkościach
2. Odczyt danych o prędkościach w ruchu swobodnym z wymuszeniem mechanizmu *broadcast*

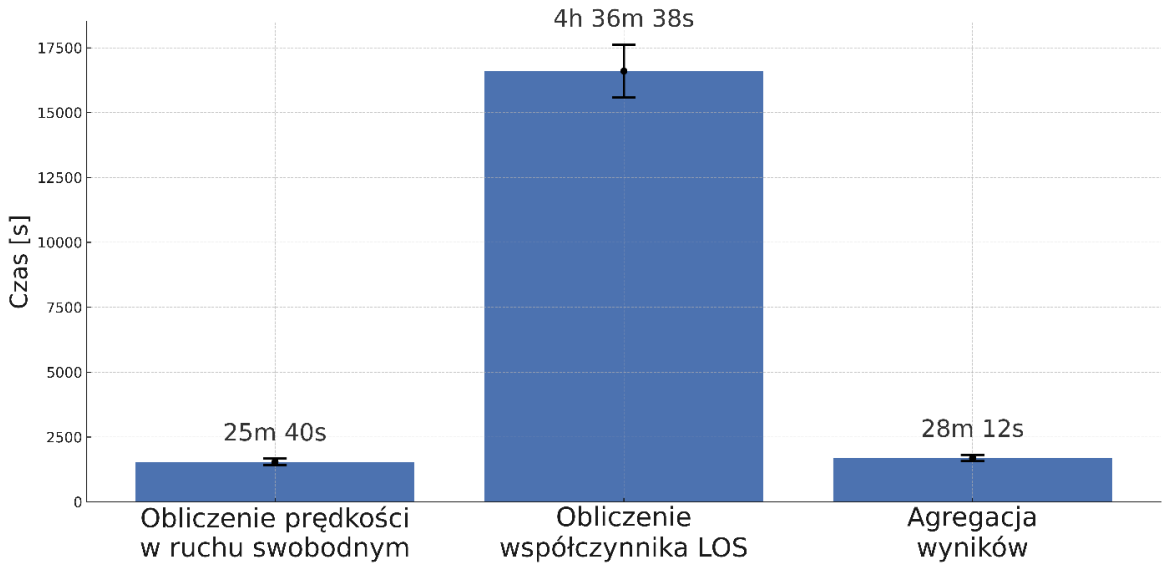
3. Złączenie lewostronne pełnego zbioru danych o prędkościach z informacją o średniej prędkości w ruchu swobodnym na podstawie identyfikatora segmentu
4. Obliczenie współczynnika LOS zgodnie ze wzorem
5. Zapis danych wyjściowych do postaci pliku Parquet

Wykonanie tych operacji zajęło średnio 4 godziny, 36 minut i 38 sekund, a wynikowy zbiór danych zajął 9.8 TB miejsca przestrzeni dyskowej.

Ostatnim etapem scenariusza było przygotowanie zbioru zagregowanego, pozwalającego na wizualizację wartości LOS w środowisku GIS. Proces składał się z następujących kroków:

1. Odczyt pełnego zbioru danych o prędkościach pojazdów z informacją o współczynniku LOS
2. Filtracja danych na podstawie kolumny z czasem pomiaru – selekcja rekordów dla danych z piątków w godzinach 16 - 18
3. Grupowanie zbioru na podstawie identyfikatora segmentu i wykonanie tabeli przestawnej ze zliczeniem liczby wystąpień każdego poziomu LOS dla danego segmentu (od A do F)
4. Obliczenie współczynnika występowania jednoznacznie negatywnych ocen LOS (od D do F) względem wszystkich dla każdego segmentu drogi
5. Przeniesienie wyników do postaci GeoDataFrame na maszynie *driver*
6. Zapis wyników do pliku SHP

Finalny zbiór danych zajął 568 MB, a czas agregacji wyniósł średnio 28 minut 12 sekund. Warto zauważyć, że w przypadku chęci zmiany logiki wyliczania współczynnika zagregowanego wystarczy zrealizować jedynie ostatni etap obliczeń (agregację danych) bez potrzeby przetwarzania całego zbioru danych od nowa. Na rysunku 81 przedstawiono podsumowanie średnich czasu trwania poszczególnych etapów realizacji scenariusza.



Rysunek 81. Zestawienie średnich czasów wykonania poszczególnych etapów scenariusza B3 ze wskazaniem odchylenia standardowego pomiędzy poszczególnymi próbami (opracowanie własne)

Scenariusz B3 stanowi bardzo dobry przykład, gdzie zastosowanie środowiska SBD może przynieść przewagę w analizie danych przestrzennych. Wejściowy, bardzo duży zbiór danych, nie byłby możliwy do prostej analizy z użyciem standardowych narzędzi. Odczyt blisko 9TB danych na raz, nawet przy zastosowaniu szybkich dysków zająłby znaczny czas, a ograniczenia pamięci RAM wymusiłyby przeprowadzanie obliczeń w rozbiciu na podzbiory. W przypadku środowiska SBD CENAGIS problemy te rozwiązywane są przez system plików HDFS umożliwiający odczyt wielu małych części całego zbioru przez wiele maszyn *worker* obsługiwanych przez Apache Spark. Występuje tutaj dodatkowo optymalna sytuacja, gdzie zbiór danych o prędkościach swobodnych podlegający złączeniu jest niewielki. Daje to możliwość zastosowania mechanizmu *broadcast* i przeniesienia go w całości na wszystkie maszyny *worker*. Optymalizuje to najbardziej czasochłonny etap przetworzeń. Wykonanie całości operacji w Scenariuszu 3 zajęło łącznie ok. 5 godzin 30 minut. Jest to czas satysfakcjonujący biorąc pod uwagę wielkość danych oraz ich zakres czasowy oraz przestrzenny.

10.4. Scenariusz B4 – mapa anomalii

Czwarty scenariusz zakłada opracowanie mapy anomalii w ruchu drogowym w oparciu o analizę wartości współczynnika LOS na sąsiadujących odcinkach sieci drogowej. Zastosowany algorytm został opisany w pracy „Detection of urban traffic patterns from Floating

Car Data (FCD)" (Altintasi, Tuydes-Yaman i Tuncay, 2017). Składa się on z dwóch głównych kroków.

Pierwszy etap polega na przypisaniu każdemu segmentowi sieci drogowej stanu ruchu na podstawie wartości współczynnika LOS, zgodnie z następującą klasyfikacją:

- LOS A lub B: stan 1
- LOS C: stan 2
- LOS D: stan 3
- LOS E lub F: stan 4

W drugim etapie, na podstawie kombinacji stanów ruchu w segmentach sąsiadujących, wyznaczany jest wzorzec ruchu zgodnie z zasadami przedstawionymi w tabeli 33.

Tabela 33. Tabelaryczna reprezentacja sposobu detekcji anomalii w ruchu drogowym w oparciu o współczynnik LOS na sąsiadujących segmentach sieci drogowej na podstawie (Altintasi, Tuydes-Yaman i Tuncay, 2017)

Wzorzec ruchu	Sekwencja stanów ruchu w sąsiadujących segmentach drogi
Swobodny przepływ	1-1
Stabilny przepływ	2-2
Przepływ przejściowy	1-2, 2-1
Zbliżający się przepływ niestabilny	3-3
Zbliżające się zatłoczenie	2-4, 1-4
Przepływ zatłoczony	4-4
Gwałtowne hamowanie	1-3

Do przedstawionej powyżej listy wzorców przepływu dodano kategorię „inne", obejmującą wszystkie kombinacje stanów przepływu, które nie zostały zaklasyfikowane do żadnego z wcześniej zdefiniowanych wzorców.

Dodatkowo, na potrzeby czytelnej wizualizacji wyników analizy, wzorce przepływu podzielono na dwie grupy:

- Wzorce pozytywne: przepływ swobodny, przepływ stabilny, przepływ przejściowy
- Wzorce negatywne: zbliżająca się destabilizacja przepływu, zbliżające się zatłoczenie, przepływ zatłoczony, gwałtowne spowolnienie

Analiza anomalii w ruchu drogowym pozwala na ekstrakcje bardzo przydatnych informacji, np. wykrycia miejsc, gdzie regularnie występują gwałtowne spowolnienia płynności ruchu).

Wyzwaniem obliczeniowym w tym scenariuszu jest konieczność wykonania złączenia danych tak, aby uzyskać informacje o wartości współczynnika LOS w tym samym punkcie czasu na sąsiadującym segmencie drogi. Tworzy to konieczność przeprowadzenia wewnętrznego złączenia bardzo dużego zbioru danych o liczności ok. 423.5 miliarda rekordów. Podobnie, jak w przypadku scenariusza B2 wykonanie takiego złączenia nie było możliwe z użyciem standardowych metod. W celu realizacji tego zadania zdecydowano się na zastosowanie analogicznego podejścia do łączenia danych jak w przypadku algorytmu rozproszonego złączenia przestrzennego. Utworzono indeks oparty o kolumnę z czasem pomiaru, gdyż partycjonowanie przestrzenne mogłoby wpłynąć na wynik analizy (dwa sąsiadujące segmenty drogi mogłyby znaleźć się w różnych partycjach przestrzennych). Po wykonaniu złączenia, również i w tym przypadku dane zagregowano do postaci możliwej do wizualizacji w oprogramowaniu GIS. Dla każdego segmentu drogi zliczono liczbę występujących dla niego anomalii każdego typu z tych możliwych do wyliczenia podczas analizy dwóch sąsiadujących segmentów drogi.

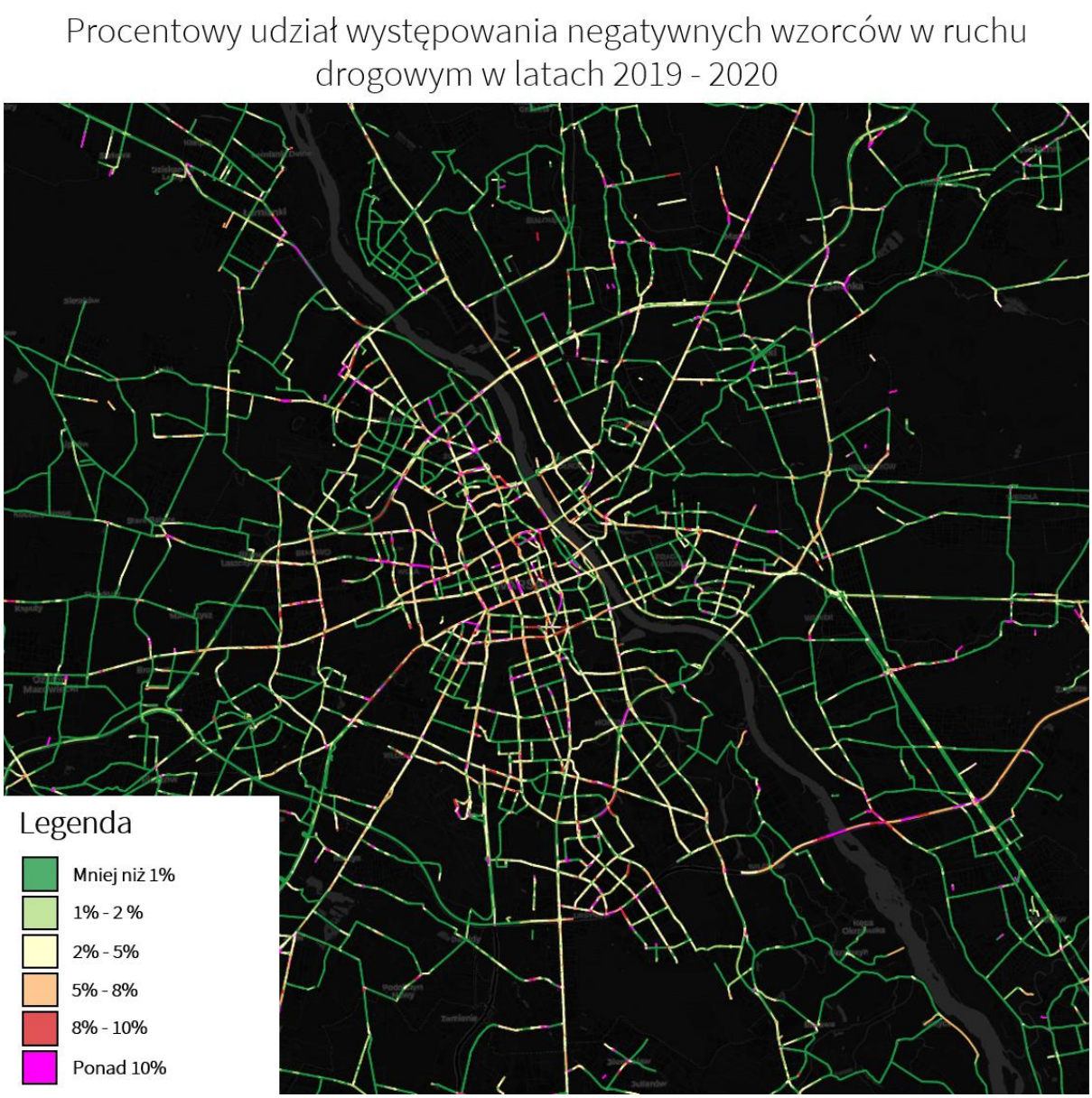
Kolejne etapy obliczeń obejmowały:

- Przygotowanie danych wejściowych. Odczytano dane o średnich prędkościach pojazdów i połączono je z wartościami prędkości w ruchu swobodnym (wykorzystano zbiór utworzony w scenariuszu B3). Wyniki zapisano w formacie Parquet, partycjonowanym według czasu pomiaru, co umożliwiło zastosowanie metody rozproszonego złączenia przedstawionej w scenariuszu B2.
- Analiza segmentów sąsiadujących. Dla każdego odcinka drogi zidentyfikowano segmenty sąsiednie i przypisano im średnie prędkości w parach segmentów.

W przypadku skrzyżowań, gdzie segmentów sąsiednich było więcej, przyjmowano średnią prędkość ze wszystkich sąsiadów. Należy mieć na uwadze, że takie uśrednianie może wprowadzać niepewność wyników w obrębie skrzyżowań. Wyniki zapisano w formacie Parquet.

- Wyznaczenie stanów ruchu i wzorców. Na podstawie prędkości obliczono współczynniki LOS dla par segmentów, a następnie przypisano każdy odcinek do jednego z ośmiu wzorców ruchu. Wyniki zapisano w formacie Parquet.
- Agregacja wyników. Zliczono częstość występowania poszczególnych wzorców dla każdego odcinka oraz obliczono udział wzorców negatywnych względem pozytywnych. Pozwoliło to określić, gdzie anomalie ruchu występują najczęściej. Zagregowane wyniki przekształcono do geopandas DataFrame i zapisano jako plik ESRI SHAPEFILE, co umożliwiło ich dalszą wizualizację.

Na rysunku 82 przedstawiono fragment wizualizacji wyniku analizy dla obszaru Warszawy. Wygenerowana mapa pokazuje procentowy udział występowania negatywnych wzorców ruchu dla poszczególnych segmentów sieci drogowej w latach 2019 - 2020. Pomarańczowa, czerwona i fioletowa barwa pokazują miejsca, w których najczęściej dochodzi do spowolnień w ruchu oraz korków.



Rysunek 82. Wizualizacja fragmentu danych wyjściowych (Warszawa i okolice) z obliczonym współczynnikiem występowania negatywnych anomalii w ruchu drogowym (opracowanie własne), podkład mapowy: OpenStreetMaps

Na tym etapie wykorzystano następujące parametry klastra:

64 x (8 vCPU, 16 GB RAM) – łącznie 512 vCPU.

Podobnie, jak w przypadku scenariusza B3, w scenariuszu B4 zbiorem wejściowym były dane o prędkościach pojazdów na odcinkach dróg. Obliczenia rozbito na cztery główne etapy:

1. Przygotowanie zbioru partycjonowanego czasowo
2. Złączenie sąsiadujących odcinków dróg
3. Klasyfikacja wzorców

4. Obliczenie współczynników podsumowujących i zapis do formatu ESRI SHAPEFILE

Przygotowanie zbioru partycjonowanego czasowo przeprowadzono wykonując następujące czynności:

1. Odczyt pełnego zbioru danych o prędkościach pojazdów.
2. Odczyt danych o prędkościach w ruchu swobodnym z wymuszeniem mechanizmu broadcast.
3. Złączenie lewostronne pełnego zbioru danych o prędkościach z informacją o średniej prędkości w ruchu swobodnym na podstawie identyfikatora segmentu.
4. Utworzenie kolumny „*par_id*" złożonej z dnia roku oraz godziny danego pomiaru.
5. Zapis danych wyjściowych do postaci pliku Parquet z partycjonowaniem plików na podstawie kolumny „*par_id*".

Kluczowym dla wydajności i dalszych obliczeń aspektem była metoda doboru sposobu partycjonowania, a więc sposób, w jaki została utworzona kolumna „*par_id*". Przyjęta metoda spowodowała utworzenie 8686 partycji. Maksymalna liczba partycji możliwa do utworzenia w przyjętym podejściu to 8784 (24 godziny pomnożone przez 365 dni dodane do 24 godzin z roku przestępnego 2020). Jednak w danych wejściowych znalazły się pojedyncze dni, gdzie ze względu na awarie po stronie aplikacji Yanosik dane nie zostały zapisane – w związku z tym różnica ta wynika właśnie z braków w oryginalnych danych. Taki podział daje średnio ok. 1.5 GB na partycje. W ramach badań możliwości optymalizacji przetworzeń próbowano dokonać obliczeń z użyciem innych podejść do podziału przestrzennego tj. z podziałem tylko na dni roku (większe partycje) oraz na dni roku, godziny oraz minuty (mniejsze partycje). W obu przypadkach jednak próby zapisu lub odczytu takiego zbioru kończyły się błędami związanymi z przepełnieniem pamięci lub spodziewany czas obliczeń znacząco przewyższał czas osiągany z wykorzystaniem podejścia opartego o dzień roku oraz godzinę. Jest to jednak parametr zależny od zbioru i dalsze eksperymenty w tym zakresie mogłyby przynieść dodatkowe korzyści w kontekście szybkości obliczeń. Ostatecznie, ta część obliczeń zajęła średnio 4 godziny, 12 minut i 57 sekund, a zbiór po podziale zajął 13.3TB miejsca na dysku (większa zajętość miejsca

względem oryginalnego zbioru jest spowodowana innym formatem partycjonowania co miało wpływ m.in. na możliwość kompresji danych w kolumnach pliku Parquet)

Przy łączeniu sąsiadujących odcinków dróg wykonano następujące czynności:

1. Odczyt podzbiorów dla poszczególnych partycji „*par_id*"
2. Złączenie wewnętrzne każdego z podzbiorów w celu uzyskania informacji o średniej prędkości i średniej prędkości w ruchu swobodnym na odcinku drogi poprzedzającym odcinek badany na podstawie kolumn „*start_node_id*" i „*end_node_id*". Złączenie przeprowadzono w taki sposób aby uzyskać odcinki dla których „*end_node_id*" jest identyczne jak „*start_node_id*" odcinka badanego. W przypadku gdy takich odcinków było więcej niż jeden wartości prędkości i prędkości w ruchu swobodnym uśredniano
3. Zapis wynikowego podzbioru do pliku Parquet z partycjonowaniem zgodnym z „*par_id*"

W tym przypadku również (podobnie jak w przypadku scenariusza B2) konieczne było zastosowanie mechanizmu współbieżnych obliczeń w celu zlecenia do Spark realizacji operacji na wielu podzbiorach na raz. W innym wypadku wykorzystanie pełnych mocy obliczeniowych klastra nie było możliwe, gdyż operacje na partycjach zajmowały zbyt mało czasu, aby wydajnie przypisać do takiej pojedynczej operacji wszystkie maszyny *worker*. W tym przypadku czas wykonania analizy na pojedynczej partycji był szybszy niż dla złączeń przestrzennych, przez co przy zastosowaniu liczby współbieżnych procesów na poziomie 64 nadal nie było możliwe równomierne rozłożenie mocy obliczeniowych maszyn *worker*, co powodowało znaczne wydłużenie czasu obliczeń do ok. 30 godzin. Przy zastosowaniu 128 procesów moc obliczeniowa maszyn *worker* została wykorzystana w pełni obniżając czas wykonania tego procesu do średnio 9 godzin, 5 minut i 21 sekund. Średni czas przygotowania kwerendy przed jej uruchomieniem w sposób rozproszony wyniósł 6 minut 39 sekund. Należy również zaznaczyć, że wykonanie tego złączenia nie było możliwe z zastosowaniem innych metod złączenia – w takim wypadku obliczenia za każdym razem kończyły się wysyceniem dostępnych zasobów dyskowych lub błędami związanymi z zerwaniem komunikacji sieciowej w klastrze. Zbiór po złączeniu zajął 25.4 TB przestrzeni dyskowej.

Trzeci krok obliczeń zakładał przeprowadzenie klasyfikacji anomalii. W tym celu należało:

1. Obliczyć współczynnik LOS dla segmentu badanego oraz segmentu/segmentów sąsiadujących
2. Na podstawie wartości obu współczynników LOS wskazać rodzaj wykrytej anomalii
3. Zapisać na dysku zbiór z dodatkową kolumną typu tekstowego z nazwą anomalii

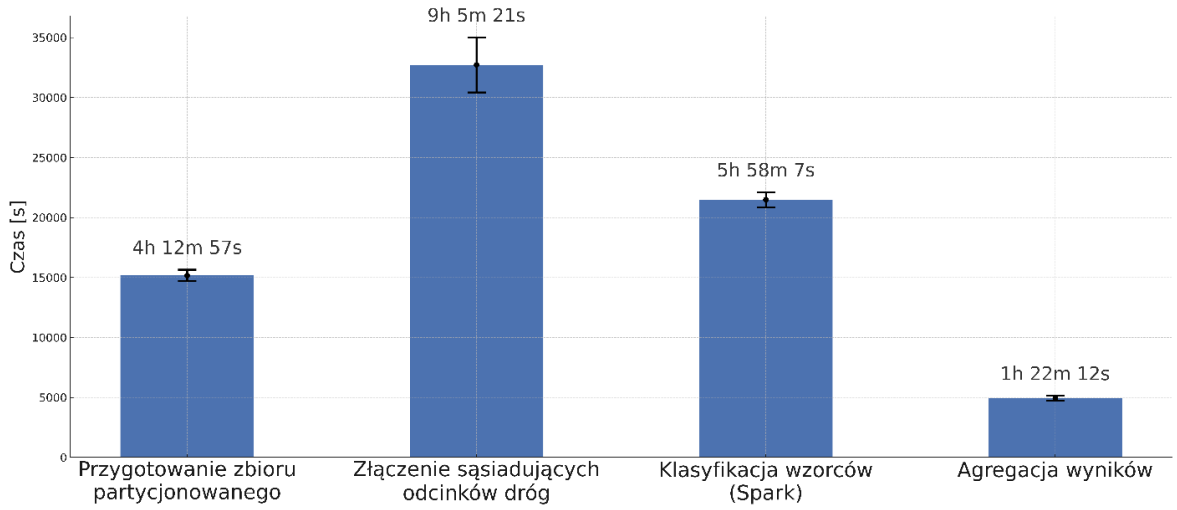
Zadanie to zrealizowano z wykorzystaniem dwóch podejść. W pierwszym z nich realizację całości oparto o funkcje pandas UDF, a w drugim o operacje wykonywane bezpośrednio z użyciem metod Spark. Badanie to wykonano w celu weryfikacji wpływu zastosowania funkcji pandas UDF na czas obliczeń w przypadkach, gdzie możliwe jest uzyskanie analogicznych wyników w oparciu o metody Spark niewymagające wykonania procesu kosztownej konwersji danych. Zgodnie z przewidywaniami operacje wykorzystujące mechanizm Pandas UDF zajęły znacznie więcej czasu. Było to średnio 17 godzin 34 minuty względem 5 godzin 58 minut z użyciem metod Spark. Wynikowy zbiór zajął 17.5 TB przestrzeni dyskowej po usunięciu kolumn przechowywujących dane częściowe do obliczeń.

W ostatnim kroku obliczeń przeprowadzono następujące operacje:

1. Zgrupowano wynikowy zbiór w grupy reprezentujące segmenty dróg
2. Dla każdego segmentu wykonano tabelę przestawną zliczającą wystąpienia poszczególnych wzorców
3. Obliczono współczynnik występowania wzorców negatywnych względem wzorców pozytywnych
4. Przeniesiono zagregowany zbiór do postaci obiektu biblioteki pandas na maszynie *driver*
5. Zapisano zbiór w formacie ESRI SHAPEFILE

Po operacji złączenia zbiór danych posiadał bardzo dużą liczbę partycji Spark (107 254), która prowadziła do występowania błędów z przepełnieniem się pamięci na etapie agregacji danych. Konieczne było więc zmniejszenie liczby partycji do 1024 co jest wystarczające dla danych zagregowanych (pojedyncza partycja nie przekroczy limitów wielkości, ani nie spowoduje przepełnienia pamięci RAM maszyn *worker*). Ta zmiana pozwoliła na poprawne wykonanie ostatniej części obliczeń. Średni czas wykonania ostatniej części scenariusza B4 wyniósł 1

godzinę 22 minuty i 12 sekund. Wynikowe pliki SHAPEFILE zajęły łącznie ok. 1GB przestrzeni dyskowej. Na rysunku 83 przedstawiono podsumowanie średnich czasu trwania poszczególnych etapów realizacji scenariusza.



Rysunek 83. Zestawienie średnich czasów wykonania poszczególnych etapów scenariusza B4 ze wskazaniem odchylenia standardowego pomiędzy poszczególnymi próbami (opracowanie własne)

Wyniki scenariusza B4 w postaci wizualizacji kartograficznych oraz wykresów przedstawiono w Załączniku nr 3.

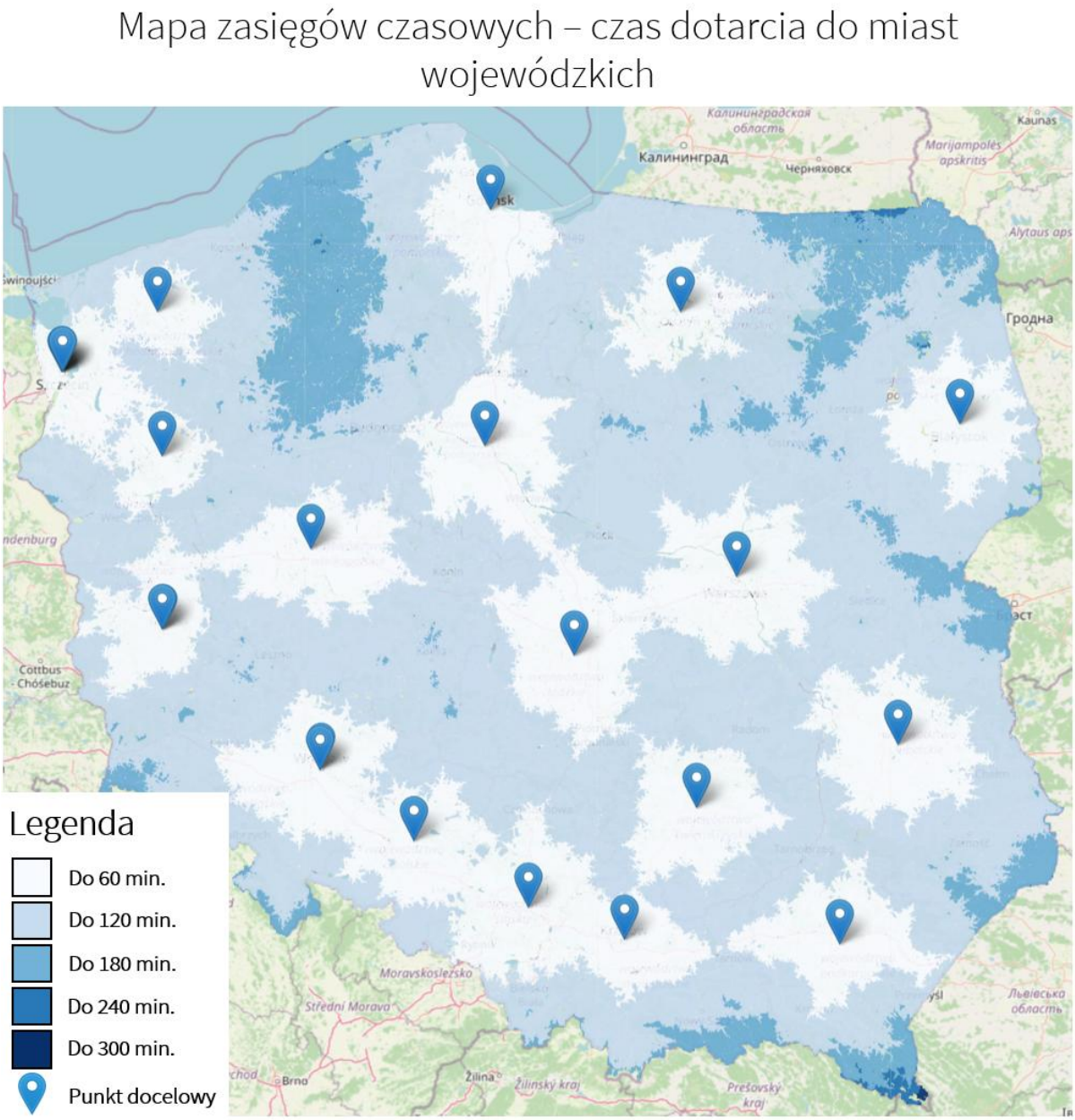
Głównym wyzwaniem scenariusza B3 było znalezienie właściwego podejścia do wewnętrznego złączenia dwóch bardzo dużych zbiorów danych. Operacja ta nie była możliwa z użyciem standardowego złączenia, a wymagała rozbicia problemu złączenia na partycje. Taka metoda pozwoliła na wykonanie koniecznych obliczeń, jednak nadal istnieje przestrzeń do jej optymalizacji w zależności od specyfiki zbioru. Istotną wadą tej metody jest również konieczność obrania takiego partycjonowania, które nie będzie miało wpływu na wynik złączenia. Przykładowo, partycjonowanie przestrzenne zastosowane w scenariuszu B2 w tym przypadku spowodowałoby potencjalne błędy w przypadku gdy sąsiadujące segmenty dróg znalazłyby się na granicach partycji przestrzennej. Realizacja scenariusza B3 wskazała także znaczne różnice w wydajności pomiędzy analizami opartymi mechanizmy konwersji danych z natywnych formatów Spark do języka Python (pandas UDF). Zgodnie z oczekiwaniami operacja ulega znacznemu spowolnieniu w wykorzystaniem mechanizmu UDF. W procesie optymalizacji obliczeń przestrzennych z użyciem tych narzędzi należy wziąć więc pod uwagę przepisania kodu języka Python do natywnych funkcji Spark.

10.5. Scenariusz B5 – interpolacja przestrzenna

Celem scenariusza było zbadanie wydajności procesu generowania rastrowych, wysokorozdzielczych map zasięgów czasowych w skali całej Polski z interpolacją czasu dojścia i uwzględnieniem przeszkód terenowych (warstwy cieki wodne i ogrodzenia trwałe z bazy danych BDOT10k). Badania wydajnościowe w tym scenariuszu skoncentrowano na etapie interpolacji przestrzennej czasu dotarcia do celu, stanowiącym końcową fazę procesu generowania wysokorozdzielczych map zasięgów czasowych.

Zagadnienie wytwarzania map zasięgów czasowych ma wiele zastosowań w dziedzinach takich jak medycyna (Weiss i inni, 2020), planowanie przestrzenne (Rossetti, Tiboni, Vetturi i Calderòn, 2015) czy transport publiczny (Vandenbulcke, Steenberghen i Thomas, 2009). Do wytwarzania map zasięgów czasowych w skali całego świata w rozdzielczości przestrzennej 1 km wykorzystywane są narzędzia SBD, w szczególności Google Earth Engine (Weiss i inni, 2020), (Weiss i inni, 2018). Zagadnienie to jest więc tematem badań z zakresu analizy dużych zbiorów danych, a także znajduje wiele praktycznych zastosowań.

Wynikiem działania algorytmu przygotowanego w ramach tego scenariusza jest mapa zasięgów czasowych (wraz z dojściem do najbliższej drogi) do wszystkich miast wojewódzkich w interwałach 60, 120, 180, 240 i 300 min. Obliczając zasięg czasowy uwzględniono zarówno czas dotarcia pieszo do drogi (dla punktów leżących w oddaleniu od dróg), jak i czas przejazdu samochodem. Brano pod uwagę przeszkody terenowe. Wizualizację wyniku obliczeń przedstawiono na rysunku 84.



Rysunek 84. Mapa zasięgów czasowych utworzona w procesie badania wydajności procesu interpolacji danych rastrowych (opracowanie własne), podkład mapowy: OpenStreetMaps

W trakcie badań sprawdzono wpływ wielkości klastra komputerowego na czas obliczeń oraz porównano wynik działania algorytmu zaimplementowanego w Spark oraz z użyciem języka Python.

Badaniu podlegał czas interpolacji przestrzennej danych. W celu uzyskania danych wejściowych do algorytmu interpolacji uprzednio wykonano proces przygotowania danych polegający na wykonaniu następujących kroków:

1. Wskazano punkty startu analizy – obszary centrów (rynek) miast wojewódzkich we wszystkich 16 województwach w Polsce.

2. Z wykorzystaniem danych o drogach z bazy danych BDOT10k utworzono graf nawigacyjny, gdzie za wagę danej krawędzi posłużył czas podróży wyznaczony w oparciu prędkość oszacowaną na podstawie klasy drogi.

3. Wyznaczenie startowych wierzchołków grafu poprzez odnalezienie wierzchołków znajdujących się najbliżej poszczególnych punktów startu.

4. Obliczenie najkrótszych tras od każdego z wierzchołków startowych do wszystkich pozostałych wierzchołków grafu znajdujących się w zakresie maksymalnie 300 minut czasu podróży.

5. Scalenie wyników w jedną warstwę reprezentującą minimalny czas dotarcia do każdego wierzchołka grafu, wyznaczoną względem wszystkich analizowanych punktów startowych.

6. Wyliczenie czasów dojazdu dla punktów rozłożonych w metrowych odstępach równomiernie na odcinkach dróg.

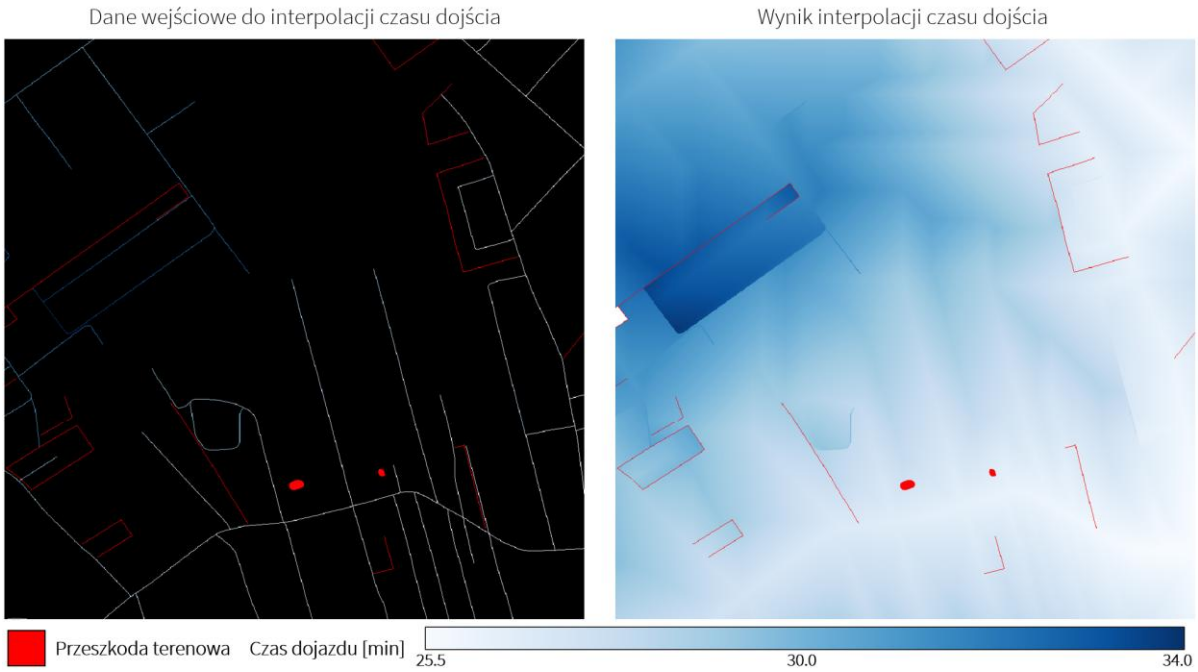
7. Przeniesienie informacji o czasie dojazdu w postaci punktów do przestrzeni rastrowej i podział na kafle o wielkości 1024 na 1024 piksele z buforem 75% (został on zastosowany w celu eliminacji późniejszych błędów związanych z interpolacją czasu dojścia na krawędziach kafli), zapis danych w kolumnie rastrowej *sum_cost.*

8. Rasteryzacja danych wektorowych z warstw cieki wodne (SKDR) i ogrodzenia trwałe (BUIB) z bazy danych BDOT10k oraz przypisanie wynikowym pikselom wartości -1, tam gdzie przeszkoda występuje i NoData tam gdzie nie występuje. Zapis danych w kolumnie rastrowej *obstacles.*

9. Zapis danych w postaci pliku Parquet na HDFS.

W wyniku opisanych powyżej operacji powstał zbiór rastrowy reprezentujący informacje o czasie dotarcia z uwzględnieniem przeszkód terenowych. Badanym w ramach tego scenariusza krokiem analizy było wyznaczenie czasu dojścia od tych pikseli, które znajdują się poza drogami (mają przypisaną wartość NoData) do najbliższego piksela reprezentującego

drogę z uwzględnieniem nieprzekraczalności przeszkód terenowych (reprezentowanych wartością -1). Na rysunku 85 przedstawiono fragment wyników przed i po procesie interpolacji.

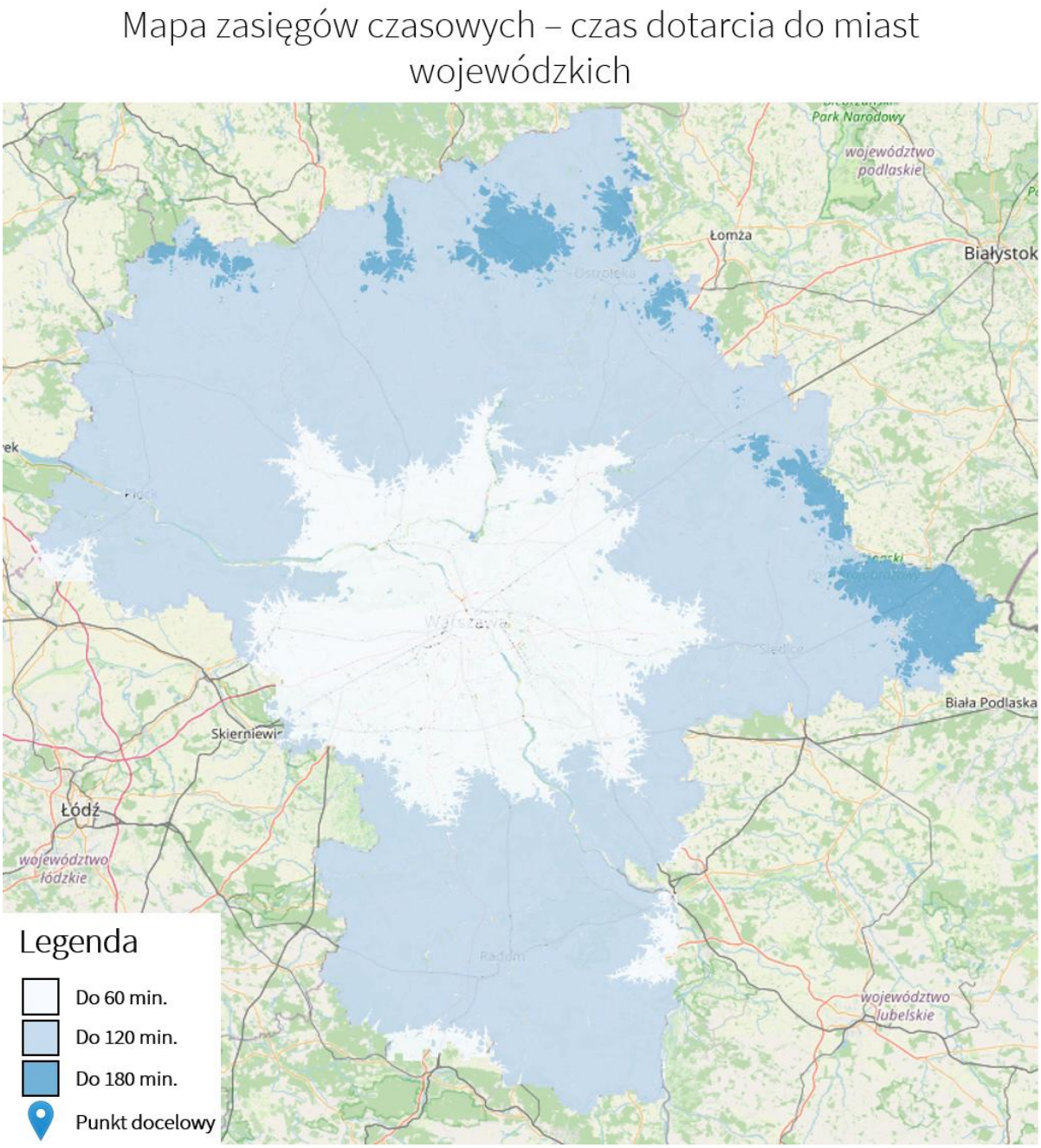


Rysunek 85. Efekt interpolacji przestrzennej czasu dojścia w procesie generowania wysokorozdzielczych map zasięgów czasowych. Po lewej stronie przedstawiono dane wejściowe do procesu interpolacji w postaci dyskretnej, a po prawej rezultat w postaci ciągłej mapy rastrowej. Kolorem czerwonym oznaczono przeszkody terenowe, a w odcieniach niebieskiego zaprezentowano czas dojazdu w minutach (opracowanie własne)

Proces ten składał się z następujących kroków:

1. Scalenie danych wejściowych. Połączono informacje o czasie dotarcia do punktów na sieci drogowej (*sum_cost*) z warstwą przeszkód terenowych (*obstacles*), tworząc wspólny zbiór danych do dalszej analizy.
2. Interpolacja czasu dojścia do najbliższego punktu sieci drogowej. Dla każdego piksela wyznaczono czas dojścia do najbliższego punktu sieci drogowej, przyjmując średnie tempo marszu 6 km/h oraz uwzględniając konieczność omijania przeszkód terenowych.
3. Zaokrąglenie wyników w górę do następujących zasięgów czasowych: 60 min, 120 min, 180 min, 240 min i 300 min.
4. Przycięcie kafli rastrowych. Ze względu na rozproszony charakter analizy, na granicach poszczególnych kafli rastrowych dochodzi do błędów w interpolacji. Z uzyskanych wyników wycięto centralne części kafli o rozmiarze 1024 × 1024 piksele, usuwając obszary buforowe.

Na potrzeby badania wydajnościowego w różnych konfiguracjach klastra przygotowano też zbiór ograniczony, reprezentujący wyniki jedynie dla województwa mazowieckiego (rysunek 86).



Rysunek 86. Fragment mapy zasięgów czasowych organiczny do obszaru województwa mazowieckiego (opracowanie własne), podkład mapowy: OpenStreetMaps

Zbiór pełny składa się z 299 017 kafli 1024x1024, a ograniczony z 34 769 takich kafli. Badania wydajnościowe na ograniczonym zbiorze przeprowadzono w następujących konfiguracjach klastra:

Tabela 34. Zestawienie parametrów klastra wykorzystane do badań wydajnościowych w scenariuszu 5B.

Suma vCPU	Suma RAM [GB]	Suma maszyn	vCPU/maszyna	RAM/maszyna [GB]
32	128	8	4	16
64	256	16	4	16
128	512	32	4	16
256	1 024	64	4	16
384	1 536	96	4	16

W ostatnim badanym scenariuszu przeprowadzono 3 badania dotyczące operacji interpolacji przestrzennej wyników algorytmu obliczania czasów dojazdu:

1. Wykonanie interpolacji na pełnym zbiorze danych
2. Wykonanie interpolacji na ograniczonym zbiorze danych (województwo mazowieckie)
3. Porównanie czasu wykonania interpolacji na ograniczonym zbiorze danych z użyciem języka Python na maszynie o odpowiadających mocach obliczeniowych

Badania interpolacji pełnego zbioru danych przeprowadzono z użyciem najmocniejszej konfiguracji z zestawienia (384 vCPU), a badania porównawcze z Python z użyciem konfiguracji najsłabszej (32 vCPU).

Zarówno w ramach Apache Spark, pakietu RasterFrames czy innych komponentów SBD CENAGIS nie istnieją narzędzia zdolne do przeprowadzenia operacji interpolacji przestrzennej danych rastrowych z uwzględnieniem przeszkód terenowych. W związku z tym konieczna była implementacja odpowiedniej logiki z użyciem funkcji pandas UDF oraz bibliotek języka Python.

Dla każdego kafelka z danymi wejściowymi do procesu interpolacji w funkcji pandas UDF wykonywano następujące czynności:

1. Konwersja do postaci macierzy NumPy
2. Nadanie przeszkodom kosztu przejścia uniemożliwiającego ich przekroczenie w analizie grafowej

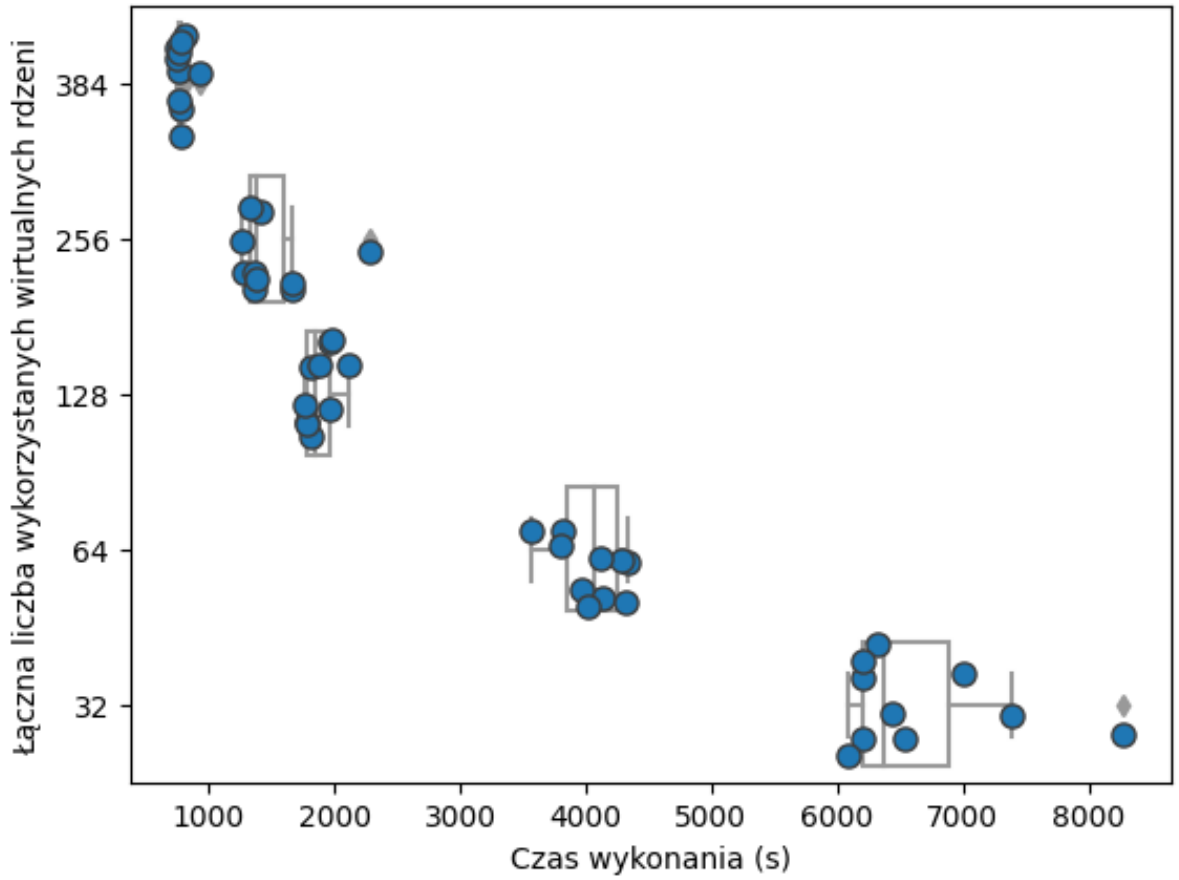
3. Przypisanie obszarom pozostałym wartości kosztu zgodnej z czasem przejścia
4. Analiza grafowa czasu dojścia do każdego punktu drogi z użyciem pakietu skimage.graph[76]
5. Przekształcenie ciągłych wartości czasu dojścia do postaci dyskretnych zakresów izochron
6. Wycięcie środka kafelka w celu eliminacji błędów związanych z obliczeniami na granicach kafelków

Wynik działania funkcji był następnie ponownie konwertowany do kafelka rastrowego RasterFrames i zapisywany na HDFS w postaci pliku Parquet. Interpolacja pełnego zbioru danych zajęła średnio 7 411 sekund (ok. 2 godziny 3 minuty).

W celu sprawdzenia wpływu wielkości mocy obliczeniowej na szybkość obliczeń przeprowadzono serie eksperymentów na danych ograniczonych do zakresu województwa mazowieckiego. To samo przetworzenie wykonano z użyciem środowiska Spark ze zmiennym parametrem sumarycznej liczby wykorzystanych vCPU, co przekłada się na liczbę wykorzystanych maszyn *worker*.

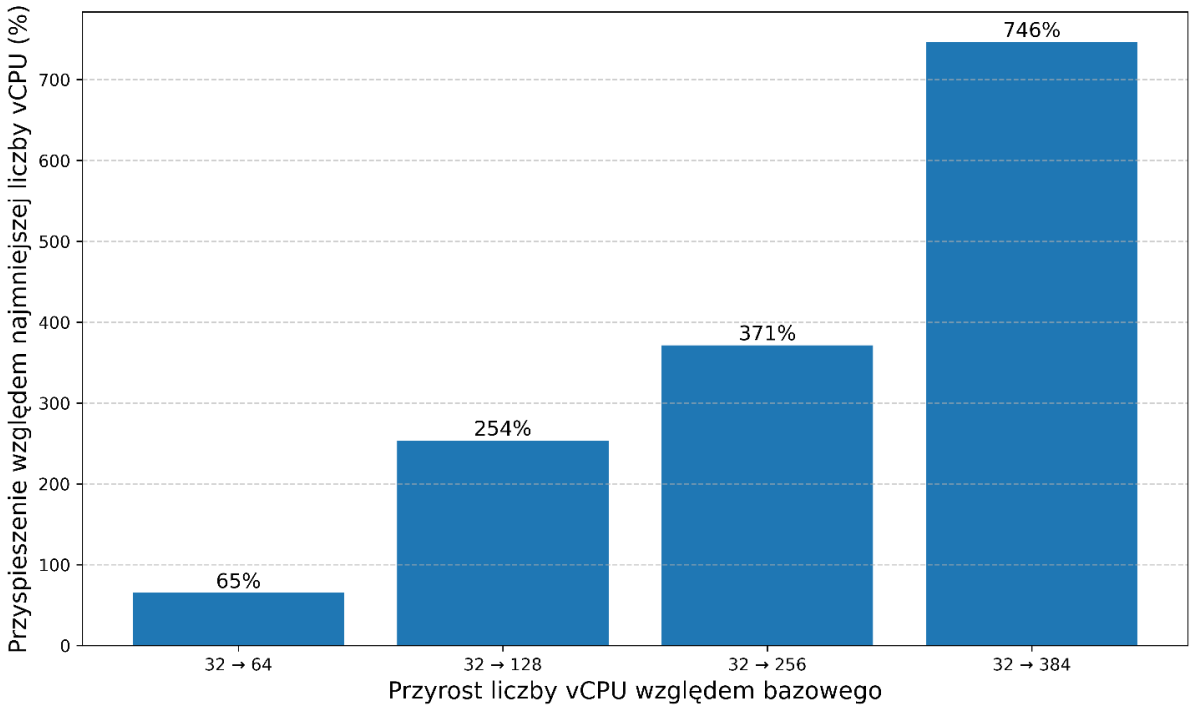
Wykonanie tej samej operacji zajęło średnio od ok. 13 minut przy zastosowaniu 384 vCPU do 1 godziny 51 minut przy zastosowaniu 32 rdzeni. Należy zauważyć nieliniowy charakter zmian w zależności od liczby rdzeni. Im więcej rdzeni tym przyspieszenie jest mniejsze w ujęciu absolutnym (rysunek 87).

[76] https://scikit-image.org/docs/stable/api/skimage.graph.html (data dostępu: 15.12.2025)



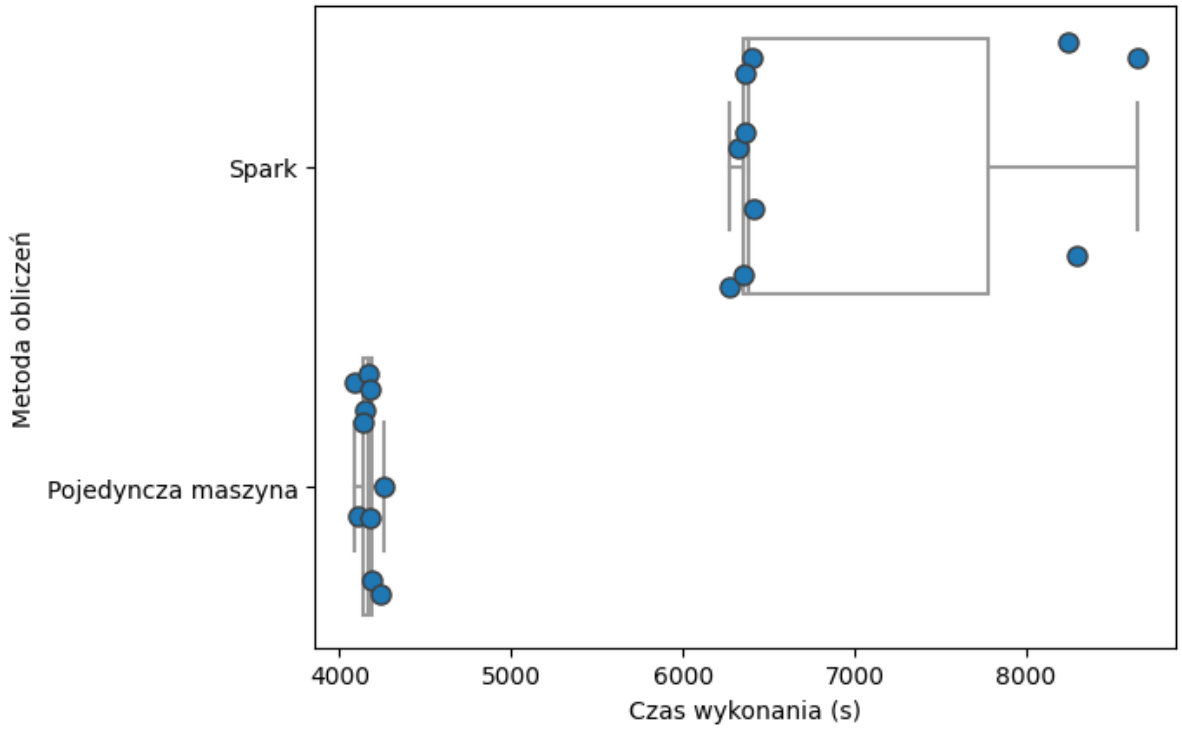
Rysunek 87. Zależność czasu wykonania zadnia interpolacji przestrzennej danych w procesie tworzenia wysokorozdzielczej mapy zasięgów czasowych w zależności od liczby wykorzystanych wirtualnych rdzeni (opracowanie własne)

W ujęciu względnym (rysunek 88) zauważyć można nieregularny sposób zmian uzyskiwanego przyspieszenia w zależności od zastosowanej liczby vCPU. Nie jest to zależność liniowa, a zwiększenie liczby vCPU o n nie powoduje n-krotnego ich przyspieszenia. Jednak różnica przy zastosowaniu 128, a 256 vCPU jest zdecydowanie mniejsza niż w przypadku 384 vCPU. Powodem takiego zjawiska może być liczba partycji lepiej zoptymalizowana pod obliczenia na większej liczbie maszyn *worker* bądź sprawniejszy odczyt danych ze względu na rozmieszczenie maszyn *worker* na większej liczbie fizycznych serwerów obliczeniowych.



Rysunek 88. Względne przyspieszenie obliczeń realizacji scenariusza B5 [%] względem konfiguracji bazowej (32 vCPU) w zależności od liczby vCPU (opracowanie własne)

Ostatni eksperyment w scenariuszu B5 polegał na sprawdzeniu różnic w wydajności pomiędzy klastrem Spark wyposażonym w 32 vCPU (8 maszyn x (4vCPU, 16 GB RAM)), a pojedynczą maszyną wyposażoną w 32 vCPU i 128 GB RAM wyposażoną w środowisko identyczne ze środowiskiem Spark i uruchamiającą te same przetworzenia skonfigurowane w taki sposób, aby wykonywały się na wszystkich dostępnych wątkach procesora na raz. Na rysunku 89 przedstawiono wyniki tego eksperymentu. **Obliczenia wykonane na pojedynczej maszynie cechują się zarówno większą szybkością, jak i stabilnością działania.** W przypadku Apache Spark zauważyć można 3 odstające pomiary. Po ich odsianiu średnia czasu realizacji obliczeń wyniosła 1 godzinę 45 minut 55 sekund. Z użyciem tych samych mocy na pojedynczej maszynie analiza zajęła średnio 1 godzinę 9 minut 28 sekund. **34% szybszy czas obliczeń wynika z brakiem opóźnień związanych z komunikacją sieciową, inicjalizacją kontekstu Spark, tworzeniem maszyn *worker* czy narzutem związanym z transformacją danych (np. partycjonowaniem).**



Rysunek 89. Porównanie czasów wykonania obliczeń interpolacji przestrzennej na klastrze Spark z 32 vCPU do czasu wykonania tych samych obliczeń z użyciem maszyny identycznej do maszyny *worker* o tych samych parametrach obliczeniowych (opracowanie własne)

Scenariusz B5 weryfikuje możliwości systemu SBD Cenagis w analizie opartej o rastry oraz analizie grafowej. Zadanie interpolacji przestrzennej punktów czasu dojazdu udało się z powodzeniem zaimplementować w środowisko CENAGIS, a możliwości skalowania środowiska rozproszonego w oparciu o Apache Spark pozwoliły na wykonanie obliczeń w skali całego kraju w stosunkowo krótkim czasie ok. 2 godzin. Przy założeniu wykorzystania mocnej stacji roboczej te obliczenia byłyby możliwe, gdyż sam rozmiar danych wyjściowych nie jest w tym przypadku przeszkodą (ok. 56 GB w formie plików Parquet). Takie obliczenia zajęłyby jednak ok. 10 godzin zakładając parametry jednostki obliczeniowej z trzeciego eksperymentu w scenariuszu B5. Biorąc pod uwagę możliwości łatwego skalowania obliczeń Apache Spark, a przez to dalszego ich przyspieszania bez konieczności dodatkowego doboru parametrów czy zmian w implementacji można uznać, że podejście to ma przewagę nad klasycznym jeżeli konieczne jest regularne (cykliczne, powtarzalne) wykonywanie tego typu analiz.

W przypadku scenariusza B5 zdecydowana większość implementacji opierała się o kod języka Python wywołany w ramach funkcji pandas UDF. Związane jest to z faktem, że narzędzia SBD nie są obecnie tak dojrzałe jak narzędzia klasyczne i nie było możliwości wykorzystania ich natywnych funkcji do osiągnięcia oczekiwanego efektu. Biblioteka RasterFrames posiada

podstawowe narzędzia do pracy z danymi rastrowymi, bez wparcia zaawansowanych technik interpolacji, a narzędzia do analizy grafowej dostępnej w ramach oprogramowania Spark, nie umożliwiały przeprowadzenia tej analizy w oparciu o dane rastrowe tak, jak jest to możliwe z wykorzystaniem biblioteki skimage. Fakt zastosowania funkcji pandas UDF niesie za sobą szereg następstw. Z jednej strony w ten sposób możliwe jest wykonanie kodu języka Python na partycjach Spark DataFrame. Może to znacznie ułatwić działanie w Spark użytkownikom z nim niezaznajomionym, ale programującym w języku Python. Ponadto rozwiązanie to pozwala na implementacje zaawansowanych algorytmów, nie dostępnych w ramach standardowych narzędzi Spark. Z drugiej jednak strony wykorzystanie funkcji pandas UDF wymaga każdorazowej transformacji danych pomiędzy kodem wykonywanym w języku Scala do tych obsługiwanych przez język Python, co wpływa na szybkość obliczeń i może powodować problemy na etapie konwersji danych (przykładowo, wersje bibliotek w środowisku SBD CENAGIS nie pozwalają na bezpośredni, zoptymalizowany transfer danych geometrycznych w ten sposób).

10.6. Podsumowanie badań wydajności

Wszystkie 5 scenariuszy badawczych udało się zrealizować z wykorzystaniem środowiska SBD CENAGIS uzyskując czasy poszczególnych kroków obliczeń na poziomie do 10 godzin w przypadku analiz uwzględniających pełne zbiory danych w oryginalnej rozdzielczości, których wielkość, w zależności od przypadku oscyluje w zakresie 3 – 10 TB w postaci przygotowanej do obliczeń rozproszonych. Potwierdza to tym samym hipotezę założoną na tym etapie badań: Zastosowanie technik obliczeń rozproszonych w środowisku *spatial big data* opartym o Apache Spark umożliwia wydajne przeprowadzenie analiz przestrzennych bardzo dużych zbiorów danych o mobilności.

Należy jednak podkreślić, że prawidłowe wdrożenie wszystkich pięciu scenariuszy wymagało przeprowadzenia wielu prób, badań różnych podejść i optymalizacji w celu uzyskania obecnych wyników. Poziom trudności implementacji rozwiązań opartych o Apache Spark i biblioteki rozszerzające jego możliwości o analizy przestrzenne jest wyższy niż w przypadku analogicznych zadań wykonywanych z użyciem desktop GIS czy języka programowania Python/R. Na ten poziom trudności składa się szereg czynników takich jak:

- Błędy środowiska – system tak złożony i innowacyjny jak środowisko SBD CENAGIS jest podatny na błędy wielu z funkcjonujących w jego ramach komponentów – systemu plików, systemu poświadczeń, infrastruktury sieciowej czy awarii poszczególnych serwerów. W trakcie realizacji badań konieczne było wielokrotne powtarzanie poszczególnych obliczeń właśnie ze względu na fakt wystąpienia błędu środowiska spowalniającego lub przerywającego obliczenia. W załączniku 4 zamieszczono najczęściej występujące błędy wraz z opisem ich przyczyn.
- Poziom dojrzałości rozwiązań – pomimo, że narzędzia SBD dostępne są już od wielu lat to nadal nie posiadają one tak szerokiego spektrum możliwości jak biblioteki języka Python czy narzędzia klasy desktop GIS. W związku z tym konieczna jest dedykowana implementacja algorytmów, które w innym wypadku byłyby dostępne w dobrze sprawdzonej i zoptymalizowanej formie. Należy pamiętać, że implementacja algorytmów możliwych do rozproszenia na wiele maszyn jest zdecydowanie bardziej skomplikowana względem klasycznych implementacji. Wpływa to w znacznym stopniu na relatywnie niską liczbę zaimplementowanych dotychczas algorytmów w tej technologii.
- Ograniczone wykorzystanie – w porównaniu do popularnych języków programowania czy oprogramowania GIS, narzędzia SBD wykorzystywane są przez stosunkowo niewielką grupę specjalistów. W związku z tym dużo trudniej jest uzyskać poradę bądź wsparcie w zakresie realizowania tego typu analiz, a dokumentacja nie jest zwykle tak bogata, jak w przypadku klasycznych narzędzi.
- Unikalność środowiska – środowiska SBD CENAGIS stanowi unikalny, eksperymentalny zbiór bibliotek rozszerzony o dodatkowe narzędzia ułatwiające prace z danymi przestrzennymi (jest to tzw. „*sandbox*"). Połączenie wielu pakietów oprogramowania w jeden spójny system wymagało wypracowania dodatkowych rozwiązań, specyficznych dla tego środowiska. Pozwala to na dostęp do bardzo szerokich możliwości w ramach jednego, spójnego systemu, ale powoduje też, że wymaga on zdobycia wiedzy dla niego specyficznej, a przez to nie tak dostępnej.
- Aktualność systemu – W związku z opisaną powyżej unikalnością środowiska dużo większym wyzwaniem jest aktualizacja poszczególnych komponentów systemu takich jak Apache Spark czy GeoMesa. Złożony zestaw bibliotek i ich wymagań powoduje, że

proste zmiany pojedynczych pakietów oprogramowania mogą być bardzo utrudnione lub niewykonalne. W związku z czym system SBD CENAGIS wykorzystuje wersje bibliotek z pewnym przesunięciem względem wersji najnowszych. Powoduje to, że niemożliwe jest wykorzystanie optymalizacji wprowadzonych przez ich twórców takich jak np. optymalizacja mechanizmu złączenia czy funkcji pandas UDF wprowadzone w ramach wersji 3. Apache Spark. Drugim efektem tego stanu rzeczy jest brak dostępu do dokumentacji powiązanej z używaną wersją oprogramowania. Część twórców udostępnia jedynie najnowszą wersję dokumentacji.

Nie mniej jednak osiągnięta wydajność z użyciem zbioru danych tego typu byłaby niemożliwa do odtworzenia z użyciem komputerów PC czy nawet wydajnych stacji roboczych, co powoduje, że narzędzia SBD mogą stanowić coraz istotniejsze narzędzie w rękach specjalistów GIS w kontekście dostępności coraz większych zbiorów danych przestrzennych i potrzebie prowadzenia coraz szerszych ich analiz.

Wydajność analizy SBD zależy od wielu czynników i parametrów, w tym tych nieuwzględnionych w ramach niniejszych badań. Jest to m.in. wpływ przyjętego partycjonowania danych, parametry środowiska Spark takie jak domyślna liczba partycji czy maksymalna wielkość partycji. Dobór tych parametrów zależy jednak od konkretnego zbioru danych i nie jest możliwe wytworzenie unikalnych zaleceń w tym zakresie.

11. Wnioski i metodyka doboru narzędzi

Stale rosnący zasób danych przestrzennych oraz dynamicznie rozwijające się technologie wymagające coraz większych mocy obliczeniowych stawiają nowe wyzwania przed specjalistami GIS w Polsce i na świecie. Ankieta przeprowadzona w ramach niniejszej pracy potwierdziła występowanie tych wyzwań – 79% ankietowanych zauważyło w swojej pracy problem ze zbyt dużą wielkością lub złożonością zbioru danych w stosunku do wykorzystywanego narzędzia. Problemy te są rozwiązywane na wiele sposobów: od zmniejszenia zakresu analizy poprzez podział obliczeń na mniejsze fragmenty po zastosowanie bardziej zaawansowanych narzędzi i algorytmów czy środowisk chmurowych. Wykorzystanie technologii SBD w środowisku specjalistów GIS w Polsce jest jednak jeszcze znikome. Przeprowadzona analiza trendów publikacyjnych oraz literatury, a także wnioski z badań porównawczych dostarczają szereg potencjalnych powodów takiego stanu rzeczy. Są one następujące:

1. Technologie SBD nie są tak popularne i przyjęte przez środowisko naukowe jak klasyczne narzędzia dekstop GIS czy języki programowania. Brakuje doświadczenia w ich wykorzystaniu.

2. Wydajne wykorzystanie tych technologii wymaga łatwego dostępu do zaawansowanych cyberinfrastruktur danych przestrzennych bądź komercyjnych środowisk chmurowych. Obecnie istnieje nadal niewiele takich rozwiązań.

3. Implementacja analiz przestrzennych z użyciem środowiska obliczeń rozproszonych wnosi dodatkowy poziom trudności i wymaga zrozumienia i uwzględnienia szeregu następstw związanych z wykonywaniem obliczeń na odseparowanych od siebie komputerach

4. Implementacja rozproszonej wersji analizy przestrzennej nie zawsze musi być wydajniejsza niż wersja klasyczna, nawet przy zastosowaniu większych mocy obliczeniowych, a proces jej optymalizacji jest bardziej czasochłonny i trudniejszy

5. Wdrożenie zaawansowanych analiz przestrzennych takich jak analizy sieciowe, analizy na obrazach z użyciem technik uczenia głębokiego czy zaawansowane analizy wielokryterialne

może wymagać implementacji i testów dodatkowych mechanizmów zapewniających poprawność wyników przy podejściu rozproszonym

6. Narzędzia SBD nie są tak dojrzałe jak inne narzędzia analizy GIS, przez co wymagają więcej dodatkowej pracy przy implementacji analiz. Co więcej, wsparcie ze źródeł dostępnych w Internecie takich jak dokumentacja techniczna, fora dyskusyjne czy nawet modele LLM[77] może być mniej wartościowe i precyzyjne niż w przypadku innych, bardziej standardowych i popularnych narzędzi ze względu na mniejszą grupę odbiorców tych metod

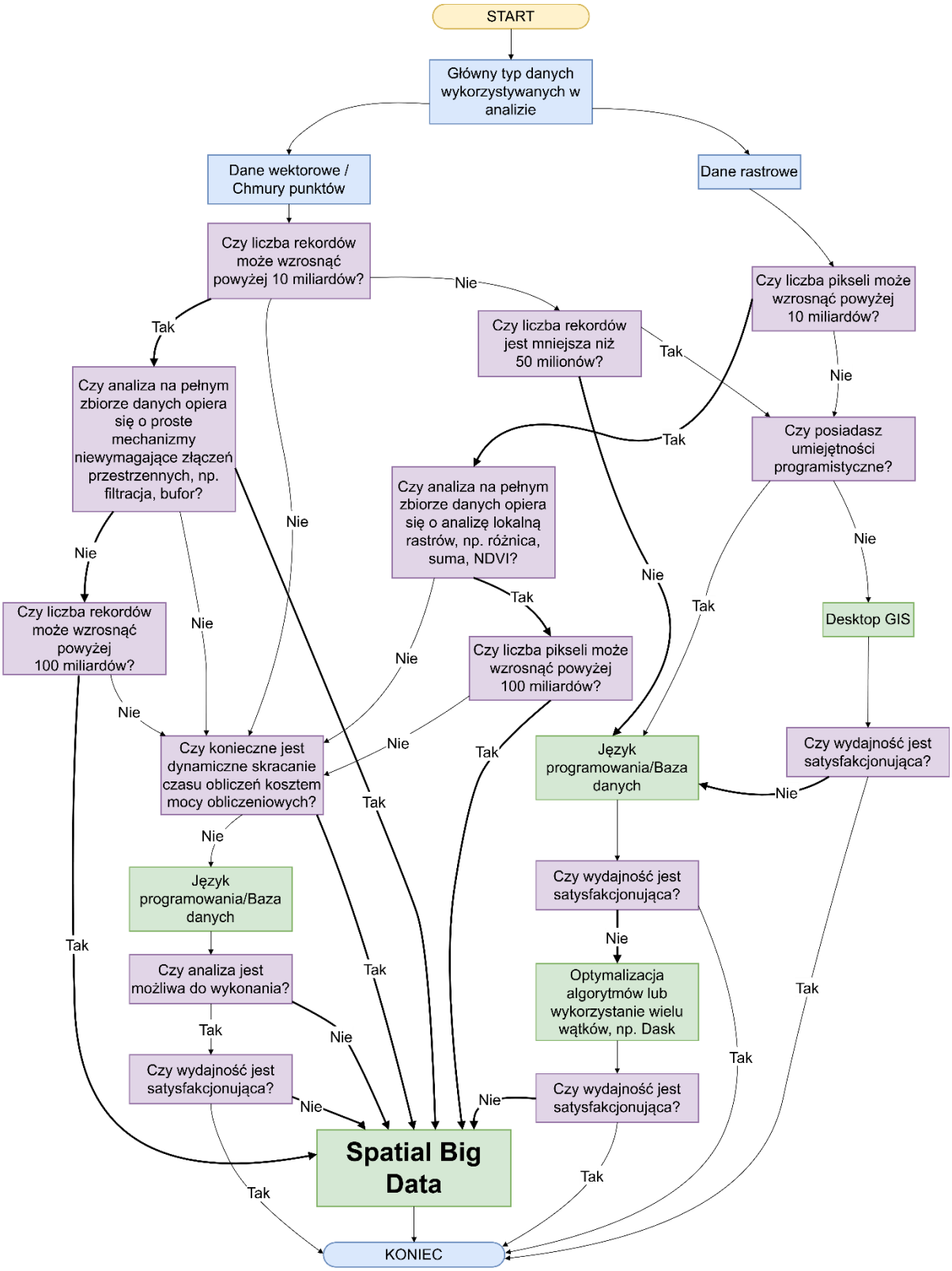
W związku z powyższymi obserwacjami wynikającymi z przeprowadzonych badań, można jasno wskazać, że narzędzia SBD, w większości wypadków, nie stanowią jeszcze bezpośredniej alternatywy dla klasycznych narzędzi GIS w szerokim spektrum zastosowań.

Badania wydajności środowiska SBD CENAGIS pozwoliły jednak wskazać obszary, gdzie zastosowanie tych narzędzi może nie tylko być podejściem optymalnym, ale też jedynym pozwalającym na wykonanie analiz przestrzennych w akceptowalnym czasie. Analizowany w ramach pracy zbiór danych FCD z aplikacji Yanosik stanowi przykład danych, których przetwarzanie z użyciem narzędzi SBD jest uzasadnione. Tak duży zasób danych, nawet przy założeniu, że istnieje możliwość umieszczenia ich w całości na dysku twardym komputera, wymaga wprowadzenia szeregu optymalizacji ich odczytu i przetwarzania w sposób iteracyjny, co znacząco komplikuje wdrożenie analizy i wydłuża jej czas. W takiej sytuacji narzędzia SBD ukazują pełnię swojego potencjału: szybki odczyt dużych danych z rozproszonego systemu plików, dynamiczny dostęp do zmiennych zasobów obliczeniowych i realizacja obliczeń w sposób rozproszony na wielu maszynach bez konieczności ręcznej implementacji tego mechanizmu. Przewaga tych narzędzi jest najlepiej widoczna w przypadkach, gdy sama analiza na dużych danych nie jest złożona i może być oparta w całości na istniejących, zoptymalizowanych narzędziach dostępnych w ramach poszczególnych pakietów oprogramowania tak, jak to miało miejsce w przypadku scenariusza B3. Scenariusze, gdzie konieczne było wdrożenie metod niedostępnych w tych pakietach np. analizy rastrowe sąsiedztwa i sieciowe w scenariuszu B5 albo złączenie przestrzenne zakresowo obejmujące cały

[77] Duże modele językowe (ang. *Large Language Models* - LLM) – modele sieci neuronowych zdolne do formułowania rozbudowanych odpowiedzi na pytania. Cechują się one bardzo duża liczbą parametrów oraz wykorzystaniem rozległych zbiorów danych na etapie uczenia.

zbiór danych (scenariusz B2) wymagają znacznie więcej testów i optymalizacji działania istniejących algorytmów w celu prawidłowego i wydajnego wykonania analizy.

Decyzja o doborze narzędzi analiz przestrzennych jest zależna od wielu czynników związanych z wielkością i charakterystyką zbioru, złożonością analizy oraz balansem pomiędzy czasem implementacji, a jej wydajnością. Nie istnieje jeden, optymalny mechanizm doboru narzędzi dla każdego przypadku, ale wnioski z przeprowadzonych prac badawczych pozwoliły na opracowanie metodyki doboru narzędzi cyberinfrastrukury CENAGIS w analizie dużych zbiorów danych przestrzennych spełniających założenia „V" *big data* (Rysunek 90 90). Badania porównawcze narzędzi wykazały, że zbiory rzędu kilkudziesięciu – kilkuset milionów punktów (scenariusze A1 i A3) czy kilkuset milionów pikseli (scenariusz A2) nie stanowią wyzwania ani dla oprogramowania klasy desktop GIS, ani dla skryptów wytworzonych w języku Python. We wszystkich przypadkach implementacje wykorzystujące język programowania były jednak wydajniejsze względem tych opartych o oprogramowanie desktop GIS. Posiadając umiejętności programistyczne lepszą decyzją będzie więc wykorzystanie języka programowania, szczególnie dla nieskomplikowanych analiz, gdzie poziom trudności implementacji jest bardzo zbliżony do wykorzystania funkcji dostępnych bezpośrednio w oprogramowaniu GIS i sprowadza się do wywołania serii prostych poleceń języka, często nawet bez wykorzystania mechanizmów takich jak pętle czy instrukcje warunkowe.



Rysunek 90. Metodyka doboru narzędzi do analizy dużych zbiorów danych przestrzennych w zależności od typu i wielkości danych oraz poziomu złożoności i celu analizy (opracowanie własne)

Scenariusz A1 badań porównawczych wykazał jednak, że nawet dla stosunkowo niewielkiego zbioru danych wykorzystanie skalowania z użyciem Apache Spark może doprowadzić do

uzyskania lepszej wydajności końcowej, kosztem wykorzystania większych mocy obliczeniowych. W tym przypadku 3-4 krotny wzrost wydajności związany był z zastosowaniem 50 krotnie większych zasobów obliczeniowych. Jednak, gdy nie jest możliwa dalsza optymalizacja lokalnych algorytmów, a wszystkie wątki procesora zostały już w pełni wykorzystane (np. z użyciem oprogramowania Dask), może to być jedyna ścieżka niewymagająca całkowitej zmiany podejścia do implementacji (tj. wykorzystując trudniejsze, ale wydajniejsze języki programowania jak język C czy implementując analizę w oparciu o jednostki GPU). Iteracyjne podejście do wdrażania rozwiązań rozpoczynające się od próby napisania własnego kodu w Pythonie, przez optymalizacje lokalną wykorzystującą wiele wątków pojedynczej maszyny, aż po wdrożenie w oparciu o techniki SBD takie jak Apache Spark jest znacznie ułatwione ze względu na wysoki stopień integracji poszczególnych rozwiązań. Przykładowo, skrypt języka Python oparty o biblioteki pandas/geopandas może być zrównoleglony z wykorzystaniem środowiska Dask bez potrzeby implementacji dodatkowej logiki. W środowisku Apache Spark z kolei mechanizm pandas UDF pozwala na wykorzystanie logiki zaprogramowanej w języku Python bezpośrednio do Spark DataFrame znacznie skracając czas implementacji. Takie rozwiązanie nie będzie, w większości wypadków, tak wydajne, jak zaprojektowane od początku z myślą o Apache Spark, ale pozwoli na szybkie skalowanie rozwiązania na wiele maszyn obliczeniowych i ułatwi decyzje o potrzebie dalszej jego optymalizacji.

Zalety zastosowania technik SBD ujawniają się jednak przy znacznie wyższym wolumenie danych. Wykorzystany podczas realizacji rozprawy doktorskiej zbiór danych stanowiący 1/10 całego zbioru danych pomiarowych FCD dla Polski zawiera ok. 10 miliardów punktów zajmując blisko 500GB w formie skompresowanego pliku Parquet. Tak duży zbiór (choć jest tylko 1/10 całości) nie jest już możliwy do załadowania do pamięci RAM komputera w całości, nawet w przypadku zastosowania zaawansowanej stacji roboczej. Przy założeniu rozbicia tego zbioru na fragmenty wielkości zbioru analizowanego w scenariuszu A1 wykonanie eksperymentu zajęłoby ok. 22 godziny nie licząc czasu podziału zbioru na fragmenty. Natomiast zadanie to zostało zrealizowane z użyciem narzędzi SBD w czasie 2 godzin 40 minut. Przewaga metod SBD jest większa w przypadku stosunkowo prostych operacji przestrzennych możliwych do przeprowadzenia na pojedynczych rekordach bądź grupach rekordów, niewymagających

złączeń przestrzennych bądź zaawansowanej, niedostępnej we wspieranych pakietach oprogramowania algorytmiki. W szczególności złączenie przestrzenne dwóch dużych zbiorów danych stanowi wyzwanie i wymaga wdrożenia dedykowanych rozwiązań.

W przypadku danych rastrowych możliwości analityczne środowiska SBD CENAGIS są bardziej ograniczone, a sama ich specyfika powoduje większą złożoność podczas wdrożenia analizy SBD. Podział danych na kafelki wymusza implementacje dodatkowej logiki w przypadku, gdy stosowane algorytmy oparte są o relacje sąsiedztwa pomiędzy pikselami, co miało miejsce zarówno w scenariuszu A2, jak i w scenariuszu B5. W przypadku danych o wielkości rzędu kilkudziesięciu miliardów pikseli (obszar Mazowsza w scenariuszu B5) możliwe było wykonanie analizy zarówno z użyciem SBD, jak i wydajnej stacji roboczej. Skalowanie obliczeń z użyciem SBD pozwoliło na uzyskanie znacznie krótszego czasu analizy na poziomie kilkunastu minut względem 1 godziny 10 minut w przypadku analizy z wykorzystaniem pojedynczej maszyny. W przypadku braku potrzeby skalowania obliczeń, rozwiązaniem optymalnym będzie jednak pozostanie przy implementacji lokalnej (szybszej o 34% względem implementacji w Apache Spark przy zastosowaniu tych samych mocy obliczeniowych). Próba przeprowadzenia tego eksperymentu w skali całego kraju (ponad 300 miliardów pikseli) zajęłaby jednak blisko 10 godzin z użyciem pojedynczej maszyny, podczas gdy implementacja oparta o Spark pozwoliła na zakończenie obliczeń w przeciągu ok. 2 godzin. Jeżeli analiza ma jednorazowy charakter, to być może ta różnica nie jest istotna. Ale gdy analiza realizowana jest na potrzeby służb ratunkowych czy zarządzania kryzysowego, lub jeżeli istnieje potrzeba jej powtarzania lub iteracyjnej parametryzacji to wybór narzędzia może mieć krytyczne znaczenie.

Decyzja o zastosowaniu metod SBD powinna być więc poprzedzona analizą maksymalnego wolumenu danych koniecznych do obsłużenia w trakcie analizy przestrzennej. W przypadku mniejszych zbiorów zastosowanie metod SBD nie przyniesie większych korzyści, a w przypadku zbiorów rzędu dziesiątek miliardów rekordów podejściem optymalnym będzie iteracyjna implementacja analizy w języku programowania Python (bądź innym języku programowania oferującym szerokie wsparcie dla obliczeń przestrzennych np. R). W przypadku gdy standardowe techniki nie pozwolą na osiągnięcie odpowiedniej wydajności, możliwe jest proste wdrożenie optymalizacji wykorzystania lokalnych wątków procesora. Gdy to podejście uniemożliwi analizę, wdrożenie analizy w oparciu o posiadany kod (działający dla podzbioru

danych) będzie zwykle stosunkowo nieskomplikowane z użyciem Apache Spark (w szczególności w przypadku danych wektorowych/chmur punktów). Jeżeli liczba rekordów przekracza setki miliardów, a wielkość danych sięga terabajtów, narzędzia SBD mogą być optymalnym lub jedynym rozwiązaniem. Nawet jeżeli nie umożliwią wykonania pełnej analizy w prosty sposób, to podejście rozproszone pozwoli na wygodne przechowywanie i wstępną filtrację danych do dalszych analiz.

Do przykładów analiz w skali całego kraju wielkości Polski, dla których zastosowanie metod SBD jest uzasadnione można zaliczyć generowanie wysokorozdzielczych map zasięgów czasowych, szczególnie jeżeli założy się regularne uaktualnianie mapy, przykładowo na podstawie pomiarów FCD. W przypadku danych wektorowych, wdrożenie SBD może być konieczne dla systemów regularnego uaktualniania informacji o średniej prędkości na segmentach dróg w oparciu o surowe pomiary FCD. Dobrym przykładem, gdzie analizy SBD mogą być niezbędne jest też tworzenie i aktualizacja HD Maps[78]. Przeprowadzone badania potwierdzają, że kluczową cechą *spatial big data* jest prędkość przyrostu (ang. *velocity* z definicji 3V). Każdą z wymienionych powyżej analiz można jednorazowo przeprowadzić w skali kraju z użyciem wydajnej stacji roboczej. Jednak w przypadku gdy zasób danych regularnie wzrasta i istnieje konieczność regularnego wykonywania kolejnych analiz w określonym czasie zastosowanie metod SBD staje się opłacalne.

Przeprowadzone badania nie są wolne od ograniczeń. Każda cyberinfrastruktura danych przestrzennych umożliwiająca obliczenia SBD składa się z innych rozwiązań zarówno sprzętowych, jak i w kontekście oprogramowania. Nie jest więc możliwe wnioskowanie ogólnie na temat wydajności takich infrastruktur, a jedynie w odniesieniu do konkretnych przykładów, takich jak CENAGIS. W przypadku zastosowania innego sprzętu, pakietów oprogramowania czy nawet ich wersji osiągnięte wyniki mogą się różnić.

Badanie ankietowe objęło ekspertów GIS reprezentujących zarówno sektor komercyjny, naukowy oraz instytucje. Zebranie odpowiedzi pochodziły wyłącznie z Polski, co uniemożliwia

[78] Wysokorozdzielcza, wielopoziomowa aktualizowana na bieżąco mapa wykorzystywana na potrzeby nawigacji samochodów autonomicznych. Wytwarzana na podstawa wielu różnych rodzajów danych przestrzennych (wektorów, rastrów i chmur punktów). W założeniach ma służyć jako dodatkowy sensor dla pojazdów pozwalający na szerszą analizę otoczenia w porównaniu do innych rodzajów czujników (Liu, Wang i Zhang, 2020).

rozszerzenie wniosków płynących z badania na inne kraje. W oparciu o analizę literatury można jednak stwierdzić, że w ośrodkach w USA i Chinach wykorzystanie technologii SBD może różnić się od tego w Polsce.

Badania wydajnościowe nie wyczerpują zakresu możliwych do sprawdzenia parametrów, narzędzi czy podejść do rozwiązania poszczególnych problemów. Zarówno w kontekście badań porównawczych SBD i GIS oraz badań wydajności SBD możliwa jest dalsza optymalizacja algorytmów z wykorzystaniem każdego z badanych narzędzi. Nie zbadano szczegółowego wpływu parametrów tworzenia partycji przestrzennych (wielkość okna, uzależnienie wielkości okna od gęstości danych, liczba partycji Spark na jedną partycję przestrzenną) czy nie zweryfikowano innych konfiguracji złączenia przestrzennego niż punkt z poligonem w relacji „*within*". Nie przeprowadzono też badań porównujących inne szeroko wykorzystywane technologie takie jak język programowania R czy Google Earth Engine.

Nie sprawdzono potencjalnej wydajności innych metod rozpraszania obliczeń takich jak Dask czy nie zweryfikowano wpływu na wydajność zastosowania popularnego w badaniach naukowych pakietu rozszerzającego Apache Spark o możliwość analiz przestrzennych tj. Apache Sedona. Nie sprawdzono wpływu optymalizacji dostępnych w nowszych wersjach pakietu Spark takich jak dynamiczne filtrowanie partycji (ang. *Dynamic Partition Pruning*), adaptacyjne dopasowanie strategii wykonania (ang. *Adaptive Query Execution*)[79] czy optymalizacje transferu danych do funkcji pandas UDF.

Część z tych podejść technologicznych nie była wykonalna w badanym środowisku SBD, ze względu na złożony i wieloetapowy proces jego aktualizacji. Jest to kompleksowe środowisko, składające się z wielu pakietów oprogramowania typu *open source* powiązanych ze sobą w unikalny sposób z użyciem specyficznych tylko dla tej infrastruktury rozwiązań programistycznych. W dynamicznym obszarze, jakim jest SBD, wiele pakietów oprogramowania jest stale aktualizowanych i rozwijanych. Inne natomiast są wygaszane, a funkcje dostępne w ich ramach są w części przypadków przenoszone do innych pakietów. Czasami przestają być

[79] Metody optymalizacji wykonywania obliczeń w Spark wprowadzone w wersji 3 środowiska. Dynamiczne filtrowanie partycji pozwala na dynamiczne ograniczenie wielkości wczytywanych danych tylko do tych potrzebnych do wykonania operacji. Adaptacyjne dopasowanie strategii wykonania polega na modyfikacji planu zapytania na bazie rzeczywistych statystyk. Źródło: https://spark.apache.org/releases/spark-release-3-0-0.html (data dostępu 06.12.2025)

one jednak dostępne w ogóle. Co więcej, wiele pakietów oprogramowania jest powiązanych ze sobą i kompatybilnych tylko w pewnym zakresie wersji.

Aktualizacja systemu wymaga więc starannej wymiany niewspieranych pakietów, aktualizacji tych wspieranych i dodania nowych, wcześniej nieistniejących lub niekompatybilnych z resztą systemu. W związku z tym nie było możliwe zbadanie wszystkich wariantów optymalizacji dostępnych w ramach nowszych wersjach oprogramowania.

Przeprowadzone badania pozwoliły jednak na sformułowania kilku wniosków dotyczących dalszego rozwoju Platformy IT CENAGIS:

1. Udostępnienie możliwości konfiguracji dedykowanego środowiska Spark w nowszej wersji dla wybranych analiz.

   Nie zawsze potrzebny jest pełen pakiet wspieranych przez infrastrukturę narzędzi. Dla niektórych analiz wartościowe mogłoby być udostępnienie możliwości konfiguracji dedykowanego środowiska Spark w nowszej wersji, ale ze wsparciem tylko wybranych narzędzi analizy przestrzennej co zoptymalizuje proces.

2. Udostępnienie w ramach infrastruktury jej wersji rozwojowej.

   W kontekście wydajności bardzo istotna jest aktualność poszczególnych pakietów oprogramowania. Wersja stabilna środowiska mogłaby być uzupełniona o wersję eksperymentalną z dostępem do nowszych wersji pakietów oprogramowania

3. Udostępnienie większej liczba przykładów ze szczególnym uwzględnieniem często występujących problemów i błędów.

   Specyfika analiz SBD oraz CENAGIS wymaga zdobycia dodatkowej wiedzy. Środowisko wyposażone jest w dokumentacje oraz przykłady prostych operacji, jak i całych analiz SBD w formie notatników Jupyter, co stanowi dobry punkt startu do nauki. Większa liczba przykładów ze szczególnym uwzględnieniem często występujących problemów i błędów w optymalizacji analiz SBD z pewnością ułatwiłaby proces nauki i wdrożeń w tym środowisku

4. Implementacja modelu językowego z kontekstem środowiska SBD CENAGIS.

Wysoki poziom złożoności środowiska mógłby być obniżony poprzez zastosowanie modelu językowego sztucznej inteligencji z kontekstem rozszerzonym o informacje specyficzne dla środowiska SBD CENAGIS. Integracja takiego mechanizmu mogłaby znacznie obniżyć próg wejścia i przyspieszyć proces pracy w środowisku

Należy również pamiętać, że badania wydajnościowe SBD wymagają wykorzystania znacznych mocy obliczeniowych przez długi czas. Uwzględniając konieczność wielokrotnych powtórzeń pomiarów, wytwarzania rozwiązań, rozwiązywania błędów i weryfikacji wyników oszacowano moc obliczeniową wykorzystaną w ramach niniejszej rozprawy doktorskiej na ekwiwalent klastra o mocy 512 vCPU i 1024 GB RAM pracującego przez 148 dni bez przerwy, co można w przybliżeniu przełożyć na ponad 34 lata pracy pojedynczego komputera o najpopularniejszych obecnie[80] parametrach (6 CPU, 16GB RAM).

Możliwe są liczne kierunki dalszych badań w zakresie optymalizacji analiz SBD z użyciem cyberinfrastruktury danych przestrzennych. Zbadany zbiór danych stanowi zaledwie jeden z przykładów bardzo dużych danych przestrzennych dla terenu Polski. Potrzebne są dalsze badania szczegółowe w zakresie innych rodzajów danych takich jak chmury punktów, wysokorozdzielcze obrazowe dane satelitarne, lotnicze i z niskiego pułapu czy HD Maps dla pojazdów autonomicznych.

Badania przeprowadzone w ramach tej pracy nie poruszyły również tematu optymalizacji rozproszonych analiz przestrzennych z użyciem GPU. W środowisku SBD CENAGIS istnieje możliwość dostępu do jednostek GPU w ramach maszyn *worker* Spark, co pozwala na dalsze optymalizacje obliczeń, a także na rozproszenie procesu uczenia oraz inferencji głębokich sieci neuronowych z wykorzystaniem danych przestrzennych. Jest to zagadnienie obecnie bardzo dynamicznie rozwijane ale jednocześnie wymagające bardzo dużych mocy obliczeniowych.

Kolejnym interesującym kierunkiem badań jest analiza możliwości wdrożenia metod łączących analizę bardzo dużych zbiorów danych przestrzennych z wykorzystaniem środowiska CENAGIS w oparciu o przetwarzanie wsadowe w połączeniu z przetwarzaniem strumieniowym. W kontekście danych o mobilności, analiza strumieniowa w trybie rzeczywistym może być

[80] Baza danych parametrów komputerów osobistych użytkowników platformy Steam. Jedna z największych baz danych o sprzęcie komputerowym oparta o dane użytkowników, a nie deklaracje (https://store.steampowered.com/hwsurvey/, data dostępu 29.11.2025)

bardzo przydatna z punktu widzenia różnych aplikacji. Nie istnieje obecnie wiele narzędzi wspierających analizę strumieniową dużych danych przestrzennych i badania w tym zakresie mogłyby prowadzić do dalszych użytecznych wniosków w kontekście rozwoju narzędzi SBD.

12. Podsumowanie

Niniejsza rozprawa doktorska podejmuje problem analizy i przetwarzania dużych zbiorów danych przestrzennych z wykorzystaniem narzędzi klasy *spatial big data* wchodzących w skład Platformy IT CENAGIS, z możliwością uogólnienia wniosków. Głównym celem pracy było opracowanie zaleceń umożliwiających właściwy dobór narzędzi analizy danych przestrzennych z uwzględnieniem wielkości zbioru, charakteru danych, rodzaju wykonywanej analizy oraz kosztu implementacji, ze szczególnym naciskiem na zbiory danych typu SBD.

W pracy przeprowadzono serię eksperymentów badawczych obejmujących różnorodne scenariusze analityczne w rozbiciu na dwa główne etapy. Pierwszy z nich miał na celu określenie wolumenu zbiorów, dla których zastosowanie narzędzi GIS staje się nieoptymalne i należy rozważyć wykorzystanie metod SBD. Drugi etap miał na celu określenie możliwości i ograniczeń badanego środowiska SBD w kontekście analiz w skali całego kraju z użyciem bardzo dużych zbiorów danych przestrzennych.

W wyniku przeprowadzonych badań zidentyfikowano typy i wolumeny danych, gdzie zastosowanie metod SBD jest uzasadnione względem innych rozwiązań technologicznych. Określono także ich ograniczenia oraz przeanalizowano złożoność implementacji analiz w porównaniu do innych metod. Istotnym rezultatem pracy jest opracowanie metodyki doboru narzędzi analiz przestrzennych dużych zbiorów danych przedstawionej schematycznie na rysunku 90. Uwzględnia ona kluczowe zmienne doboru narzędzi: rodzaj analizy, charakterystykę danych, złożoność operacji przestrzennych, a także wymagania dotyczące wydajności przetworzeń. Praca z danymi SBD cechuje się dużą czasochłonnością, więc świadomy i uzasadniony dobór narzędzi analiz przestrzennych jest kluczowy zarówno w kontekście badawczym, jak i komercyjnym.

Warto podkreślić, że wnioski z przeprowadzonych badań zostaną wykorzystane do aktualizacji Podsystemu Big Data w infrastrukturze CENAGIS, a przeprowadzone testy wydajnościowe mogą zostać wykorzystane do weryfikacji poprawności działania systemu po aktualizacji i optymalizacji analiz z użyciem narzędzi niedostępnych w obecnej wersji systemu. Spójny, praktyczny zestaw scenariuszy może zarówno ułatwić proces aktualizacji i testów, ale także

może stanowić podstawę do dalszych badań w zakresie optymalizacji działania cyberinfrasuktury danych przestrzennych.

Należy jednak zauważyć, że przeprowadzone badania posiadają pewne ograniczenia. Dotyczą one w szczególności zakresu analizowanych scenariuszy, dostępności danych wejściowych oraz zależności wyników od zastosowanej infrastruktury obliczeniowej. W przyszłości zasadne jest rozszerzenie badań o dodatkowe typy danych (np. HD Maps), a także przeprowadzenie analiz z użyciem innych zestawów narzędzi SBD, takich jak Apache Sedona.

Podsumowując, rozprawa wskazuje, że kluczowym czynnikiem wpływającym na efektywność analiz przestrzennych jest dopasowanie metod przetwarzania do skali danych, ich struktury oraz złożoności operacji analitycznych. Zaproponowana metodyka stanowi próbę usystematyzowania tego procesu, umożliwiając bardziej świadome podejmowanie decyzji technologicznych w analizach dużych zbiorów danych przestrzennych. Wyniki pracy mogą stanowić podstawę do dalszych badań nad optymalizacją środowisk SBD i ich zastosowań w praktyce.

Bibliografia

Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X. i Saltz, J. (2013). Hadoop-GIS: A high performance spatial data warehousing system over MapReduce. *Proceedings of the VLDB endowment international conference on very large data bases.*

Alam, M. M. (2019). *Parallel and In-Memory Big Spatial Data Processing Systems and Benchmarking.* The University of New Brunswick.

Alam, M., Torgo, L. i Bifet, A. (2022). A Survey on Spatio-Temporal Data Analytics Systems. *ACM Computing Surveys*.

Altintasi, O., Tuydes-Yaman, H. i Tuncay, K. (2017). Detection of urban traffic patterns from Floating Car Data (FCD). *Transportation Research Procedia*, strony 382-391.

Ambros, J. i Kyselý, M. (2016). Free-flow vs car-following speeds: Does the difference matter? *Advances in Transportation Studies*, strony 17-26.

Baig, F., Teng, D., Kong, J. i Wang, F. (2021). SPEAR: Dynamic Spatio-Temporal Query Processing over High Velocity Data Streams. *Proceedings - International Conference on Data Engineering*, (strony 2279-2284).

Bao, Y., Huang, Z., Gong, X., Zhang, Y., Yin, G. i Wang, H. (2023). Optimizing segmented trajectory data storage with HBase for improved spatio-temporal query efficiency. *International Journal of Digital Earth*, strony 1124-1143.

Brodsky, I. (2018). *H3: Uber's Hexagonal Hierarchical Spatial Index*. Pobrano z lokalizacji https://www.uber.com/en-PL/blog/h3

Cao, S., Weng, Q., Du, M., Li, B., Zhong, R. i & Mo, Y. (2020). Multi-scale three-dimensional detection of urban buildings using aerial LiDAR data. *GIScience & Remote Sensing*, strony 1125-1143.

Choromański, K., Gotlib, D. i Łobodecki, J. (2023). Jupyter Technology as a Cartographer's Work Environment: A Case Study. *Proceedings of the ICA.* Kapsztad: International Cartographic Association.

Clementini, E. i Di Felice, P. (1996). A model for representing topological relationships between complex geometric features in spatial databases. *Information sciences*, strony 121-136.

Cover, T. i Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 21-27.

Cudré-Mauroux, P., Kimura, H., Lim, K. T., Rogers, J., Simakov, R., Soroush, E. i Zdonik, S. (2009). A demonstration of SciDB: a science-oriented DBMS. *Proceedings of the VLDB Endowment*, (strony 1534-1537).

Dean, J. i Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, (strony 137-150). San Francisco.

Deibe, D., Amor, M. i Doallo, R. (2020). Big data geospatial processing for massive aerial LiDAR datasets. *Remote Sensing*.

Dempsey, C. (2019). *Geography Realm*. Pobrano z lokalizacji Survey of GIS Professionals: https://www.geographyrealm.com/survey-of-gis-professionals

Departament Transportu USA. (2021). *Evolving Use of Level of Service Metrics in Transportation Analysis – Introduction.*

Deutsch, P. (1996). *DEFLATE compressed data format specification version 1.3.* Pobrano z lokalizacji https://www.rfc-editor.org/rfc/rfc1951#section-Abstract

Dritsas, E. i Trigka, M. (2025). Remote Sensing and Geospatial Analysis in the Big Data Era: A Survey. *Remote Sensing*.

Eldawy, A. i Mokbel, M. F. (2015). Spatialhadoop: A mapreduce framework for spatial data. *IEEE 31st international conference on Data Engineering*, (strony 1352-1363).

Emmanuel, I. i Stanier, C. (2016). Defining big data. *Proceedings of the International Conference on big data and advanced Wireless technologies*, strony 1-6.

Esch, T., Heldens, W., Hirner, A., Keil, M., Marconcini, M., Roth, . . . Strano, E. (2017). Breaking new ground in mapping human settlements from space–The Global Urban Footprint. *ISPRS Journal of Photogrammetry and Remote Sensing*, strony 30-42.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D. i Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 18-27.

Gotlib, D., Choromański, K. i Kaczałek, B. (2022). CENAGIS geo-cyberinfrastructure as a R&D environment for a Surveyor 4.0. *FIG Congress 2022.* Warszawa.

Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. *ACM SIGMOD Record*, strony 47-57.

Han, X., Gstrein, O. J. i Andrikopoulos, V. (2024). When we talk about Big Data, What do we really mean? Toward a more precise definition of Big Data. *Frontiers in Big Data*.

Hansen, M. C., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S. A., Tyukavina, A., . . . Townshend, J. R. (2013). High-Resolution Global Maps of 21st-Century Forest Cover Change. *Science*, strony 850-853.

Haynes, D., Mitchell, P. i Shook, E. (2020). Developing the Raster Big Data Benchmark: A Comparison of Raster Analysis on Big Data Platforms. *ISPRS International Journal of Geo-Information*.

Hu, Z., Schnase, F. L., Duffy, J. L., Lee, D. Q., Bowen, T. K. i Yang, M. K. (2017). A spatiotemporal indexing approach for efficient processing of big array-based climate data with MapReduce. *International Journal of Geographical Information Science*, strony 17-35.

Huang, Z., Chen, Y., Wan, L. i Peng, X. (2017). GeoSpark SQL: An effective framework enabling spatial queries on spark. *ISPRS International Journal of Geo-Information*.

Hughes, J. N., Annex, A., Eichelberger, C. N., Fox, A. H. i Ronquest, M. (2015). Geomesa: a distributed architecture for spatio-temporal fusion. *Geospatial informatics, fusion, and motion video analytics*, (strony 128-140).

Ishwarappa, K. i Anuradha, J. (2015). A brief introduction on big data 5Vs characteristics and hadoop technology. *Procedia Computer Science*, strony 319-324.

Ivanov, T. i Pergolesi, M. (2020). The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet. *Concurrency and Computation: Practice and Experience*.

Kałuski, M., Hościłło, A. i Gurdak, R. (2018). Accuracy assessment of the Copernicus Buildings Height 2012 layer based on the city of Warsaw. *Geoinformation Issues*, strony 53-64.

Kazemzadeh, K. i Ronchi, E. (2022). From bike to electric bike level-of-service. *Transport reviews*, strony 6-31.

Kim, S., Hoang, Y., Yu, T. T. i Kanwar, Y. S. (2023). GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases. *Big Data Research*.

Kini, A. i Emanuele, R. (2014). Geotrellis: Adding Geospatial Capabilities to Spark. *Spark Summit 2014.*

Kissling, W. D., Shi, Y., Koma, Z., Meijer, C., Ku, O., Nattino, F., . . . Grootes, M. W. (2023). Country-wide data of ecosystem structure from the third Dutch airborne laser scanning survey. *Data in Brief*.

Konieczny, B. (2022). *Predicate pushdown, why it doesn't work every time?* Pobrano z lokalizacji https://www.waitingforcode.com/apache-spark-sql/predicate-pushdown-why-doesnt-work-every-time

Kroodsma, D., Mayorga, J., Hochberg, T., Miller, N., Boerder, K., Ferretti, F., . . . Worm, B. (2018). Tracking the global footprint of fisheries. *Science*, strony 904-908.

Kurczyński, Z. i Bakuła, K. (2013). Generowanie referencyjnego numerycznego modelu terenu o zasięgu krajowym w oparciu o lotnicze skanowanie laserowe w projekcie ISOK. *Archiwum Fotogrametrii, Kartografii i Teledetekcji*, strony 59-68.

Lakshman, A. i Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS operating systems review*, strony 35-40.

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. *META group research note 6.70*.

Li, K., Wan, G., Cheng, G., Meng, L. i Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS journal of photogrammetry and remote sensing*, 269-307.

Li, X., Yu, H., Yuan, L. i Qin, X. (2022). Query Optimization for Distributed Spatio-Temporal Sensing Data Processing. *Sensors*, strony 1-21.

Litman, T. (2003). Measuring transportation. *Traffic, mobility and accessibility. ITE Journal*.

Liu, M., Wang, P., Hu, K., Gu, C., Jin, S. i Chen, L. (2024). A Method for Extracting High-Resolution Building Height Information in Rural Areas Using GF-7 Data. *Sensors*.

Liu, R., Wang, J. i Zhang, B. (2020). High Definition Map for Automated Driving: Overview and Analysis. *The Journal of Navigation*, strony 324-341.

Loukili, Y., Lakhrissi, Y. i Ali, S. E. (2022). Geospatial Big Data Platforms: A Comprehensive Review. *Journal of Cartography and Geographic Information*, strony 293-308.

Matei, Z., Mosharaf, C., Michael, J. F., Scott, S. i Ion, S. (2010). Spark: Cluster Computing with Working Sets. *HotCloud 2010.* Boston.

Michael, A., Reynold S., X., Cheng, L., Yin, H., Davies, L., Joseph K., B., . . . Matei, Z. (2015). Spark SQL: Relational Data Processing in Spark.

Morton, G. M. (1966). *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing.* Ottawa: IBM Ltd.

NVIDIA. (2023). *RAPIDS: Libraries for End to End GPU Data Science*. Pobrano z lokalizacji https://rapids.ai

Olasz, A., Thai Nguyen, B., Giachetta, R. i Kristóf, D. (2016). Big Geospatial Data processing in the IQmulus Cloud. *International Symposium on Applied Informatics and Related Areas*, (strony 68-74).

Open Geospatial Consortium. (2008). *LAS Specification 1.2.*

Pandey, V., Kipf, A., Neumann, T. i Kemper, A. (2018). How good are modern spatial analytics systems? *Proceedings of the VLDB Endowment*, (strony 1661-1673).

Pekel, J. F., Cottam, A., Gorelick, N. i Belward, A. S. (2016). High-resolution mapping of global surface water and its long-term changes. *Nature*, strony 418-422.

Pentzold, C. i Knorr, C. (2024). When data became big: revisiting the rise of an obsolete keyword. *Information, Communication & Society*, strony 600-617.

Raffa, M., Reder, A., Marras, G. F., Mancini, M., Scipione, G., Santini, M. i Mercogliano, P. (2021). VHR-REA_IT dataset: Very high resolution dynamical downscaling of ERA5 reanalysis over Italy by COSMO-CLM. *Data*.

Rahman, M. T. (2014). Extraction of urban building heights from LiDAR data: an integrated remote sensing and GIS approach. *EARSeL eProceedings*, (strony 59-64).

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. *SciPy*, strony 126-132.

Rossetti, S., Tiboni, M., Vetturi, D. i Calderòn, E. J. (2015). Pedestrian mobility and accessibility planning: some remarks towards the implementation of travel time maps. *CSE Journal*, strony 67-78.

Ruzicka, J., Tichy, T. i Hajciarova, E. (2022). Big Data Application for Urban Transport Solutions. *Smart Cities Symposium Prague, SCSP 2022*, (strony 1-3). Praga.

Sakr, M., Ray, C. i Renso, C. (2022). Big mobility data analytics: recent advances and open problems. *GeoInformatica*, strony 541-549.

Sen, R., Farris, A. i Guerra, P. (2014). *Benchmarking the Apache Accumulo Distributed Key–Value Store.*

Shafer, T. (2017). *The 42 V's of Big Data and Data Science* . Pobrano z lokalizacji https://www.kdnuggets.com/2017/04/42-vsbig-data-data-science.html

Shin, H., Lee, K. i Kwon, H. Y. (2022). A comparative experimental study of distributed storage engines for big spatial data processing using GeoSpark. *Journal of Supercomputing*, strony 2556–2579.

Shvachko, K., Kuang, H., Radia, S. i Chansler, R. (2010). The hadoop distributed file system. *IEEE 26th symposium on mass storage systems and technologies*, (strony 1-10).

Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, strony 72-101.

Tamiminia, H., Salehi, B., Mahdianpari, M., Quackenbush, L., Adeli, S. i Brisco, B. (2020). Google Earth Engine for geo-big data applications: A meta-analysis and systematic review. *ISPRS Journal of Photogrammetry and Remote Sensing, 164*, strony 152-170.

Tang, M., Yu, Y., Mahmood, A. R., Malluhi, Q. M., Ouzzani, M. i Aref, W. G. (2020). Locationspark: In-memory distributed spatial query processing and optimization. *Frontiers in Big Data.*

Tang, W., Zheng, M., Z. X., Shi, J., Yang, J. i Trettin, C. C. (2018). Big geospatial data analytics for global mangrove biomass and carbon estimation. *Sustainability*, strony 1-17.

Teijeiro Paredes, D., Amor López, M., Buján, S., R. R. i Döllner, J. (2024). Multistage strategy for ground point filtering on large-scale datasets. *Journal of Supercomputing*, strony 25974–26001.

Transportation Research Board. (1965). *Higway Capacity Manual.*

Transportation Research Board. (1994). *Higway Capacity Manual.* Wasgington DC.

Vandenbulcke, G., Steenberghen, T. i Thomas, I. (2009). Mapping accessibility in Belgium: a tool for land-use and transport planning? *Journal of Transport Geography*, strony 39-53.

Wang, M., Mao, B., Yang, Y., Shi, R. i Huang, J. (2022). Determining the level of service scale of public transport system considering the distribution of service quality. *Journal of Advanced Transportation*.

Weiss, D. J., Nelson, A., Gibson, H. S., Temperley, W., Peedell, S., Lieber, A. i Gething, P. W. (2018). A global map of travel time to cities to assess inequalities in accessibility in 2015. *Nature*, strony 333-336.

Weiss, D. J., Nelson, A., Vargas-Ruiz, C. A., Gligorić, K., Bavadekar, S., Gabrilovich, E. i Gething, P. W. (2020). Global maps of travel time to healthcare facilities. *Nature medicine*, strony 1835-1838.

Welch, T. (1984). A technique for high-performance data compression. *Computer*, strony 8-19.

Wendland, A., Gotlib, D., Choromański, K., Kaczałek, B. i Stawowski, M. (2024). Detection service based on coherent point cloud analysis with AI methods using CENAGIS libraries. *Abstracts of the ICA.*

Wielechowski, M., Czech, K. i Grzęda, Ł. (2020). Decline in Mobility: Public Transport in Poland in the time of the COVID-19 Pandemic. *Economies*.

Wu, J., Gan, W., Chao, H.-C. i Yu, P. S. (2024). Geospatial Big Data: Survey and Challenges.

Xia, G. S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J. i Zhang, L. (2018). DOTA: A large-scale dataset for object detection in aerial images. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (strony 3974-3983).

Xie, D., Li, F., Yao, B., Li, G., Chen, Z., Zhou, L. i Guo, M. (2016). Simba: spatial in-memory big data analysis. *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, (strony 1-4).

Yang, C., Raskin, R., Goodchild, M. i Gahegan, M. (2010). Geospatial Cyberinfrastructure: Past, present and future. *Computers, Environment and Urban Systems*, 264-277.

Yu, J., Wu, J. i Sarwat, M. (2015). Geospark: A cluster computing framework for processing large-scale spatial data. *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, (strony 1-4).

Zeidan, A. i Vo, H. T. (2022). Efficient spatial data partitioning for distributed k NN joins. *Journal of Big Data*.

Zeng, X., Hui, Y., Shen, J., Pavlo, A., McKinney, W. i Zhang, H. (2023). An empirical evaluation of columnar storage formats. *Proceedings of the VLDB Endowment*, (strony 148-161).

Zhang, F., Zhou, J., Liu, R., Du, Z. i Ye, X. (2016). A new design of high-performance large-scale GIS computing at a finer spatial granularity: A case study of spatial join with spark for sustainability. *Sustainability*.

Zhang, H., Du, M., Huang, W., Ding, L., Tang, D. i Jiang, J. (2022). Research of Vector Tile Construction Technology Based on Apache Sedona. *International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, strony 639-644.

Zhang, P., Stewart, K. i Li, Y. (2023). Estimating traffic speed and speeding using passively collected big mobility data and a distributed computing framework. *Transactions in GIS*, strony 1124-1144.

Zhang, Z. i Song, Y. (2024). Spatial Big Data and Analysis Strategies Supporting Geographic Information System for Transportation (GIS-T) in Conceptual Design, Modelling, and Decision-making: A Review. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, strony 461-468.

Spis rysunków

Spis tabel

Załączniki

Załącznik 1 – konfiguracja środowiska Spark

Na poniższych tabelach przedstawiono zmienne konfiguracyjne środowiska Spark wykorzystane w trakcie badań w niniejsze pracy. Część wartości (np. hasła, lokalizacje wrażliwych plików) została ukryta ze względu na bezpieczeństwo systemu. Część z nich jest też zależna od konkretnego scenariusza. W takim wypadku wartości te są wskazane w opisie danego scenariusza badawczego.

Konfiguracja uruchomieniowa

Tabela 35. Zestawienia informacji o konfiguracji uruchomieniowej środowiska wykorzystywanej do przeprowadzenia badań (opracowanie własne)

Nazwa	Wartość
Wersja Java	1.8.0_352 (Private Build)
Lokalizacja Java	/usr/lib/jvm/java-8-openjdk-amd64/jre
Wersja Scala	version 2.11.12

Właściwości Spark

Tabela 36. Zestawienia informacji o właściwościach klastra Spark wykorzystywanych do przeprowadzenia badań (opracowanie własne)

Nazwa	Wartość
Parquet.block.size	268435456
Parquet.enable.dictionary	true
spark.app.id	a0010bd9-eb5a-4891-bfd3-04f9d447a2b8-0445
spark.app.name	Mobility segments analysis
spark.cores.max	(zależne od scenariusza)
spark.default.parallelism	2048
spark.driver.host	(ukryte)
spark.driver.maxResultSize	0
spark.driver.memory	20g

Nazwa	Wartość
spark.driver.port	(ukryte)
spark.dynamicAllocation.enabled	true
spark.dynamicAllocation.initialExecutors	1
spark.dynamicAllocation.minExecutors	1
spark.eventLog.dir	/tmp
spark.eventLog.enabled	true
spark.executor.cores	(zależne od scenariusza)
spark.executor.id	driver
spark.executor.instances	(zależne od scenariusza)
spark.executor.memory	(zależne od scenariusza)
spark.executor.memoryOverhead	600
spark.executorEnv.PYSPARK_PYTHON	/usr/bin/python3.7
spark.executorEnv.PYTHONPATH	py4j-0.10.7-src.zip:pyspark.zip:geomesa_pyspark-3.1.1.dev0+cenagis1.0.zip:pytz-2022.7.1.zip:pyrasterframes-0.9.1.zip:geopyspark.zip
spark.hadoop.yarn.resourcemanager.principal	kchoromanski
spark.jars	/usr/local/cenagis/jars/geomesa-accumulo-spark-runtime-accumulo1_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-cqengine-datastore_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-cqengine_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-fs-spark-runtime_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-fs-spark_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-spark-converter_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/geomesa-z3_2.11-3.1.2-cenagis.jar,/usr/local/cenagis/jars/cenagis-raster-assembly-1.0.jar

Nazwa	Wartość
spark.kryo.registrator	org.locationtech.geomesa.spark.GeoMesaSparkKryoRegistrator,geotrellis.spark.store.kryo.KryoRegistrator,geopyspark.geotools.kryo.ExpandedKryoRegistrator
spark.kryoserializer.buffer.max	2047m
spark.local.dir	/opt/spark_tmp
spark.locality.wait	0s
spark.master	(ukryte)
spark.mesos.containerizer	mesos
spark.mesos.executor.docker.forcePullImage	true
spark.mesos.executor.docker.image	gitlab.cenagis.local:5005/cenagis-admins/cenagis-notebook:executor
spark.mesos.executor.docker.parameters	network=host
spark.mesos.executor.docker.volumes	/etc/krb5.conf:/etc/krb5.conf:ro,/etc/krb5.keytab:/etc/krb5.keytab:ro,/var/lib/sss/pipes/:/var/lib/sss/pipes/:rw,/var/lib/sss/mc/:/var/lib/sss/mc/:ro,/opt/spark_tmp:/opt/spark_tmp,/var/lib/hadoop-hdfs:/var/lib/hadoop-hdfs,/usr/hdp:/usr/hdp:ro,/etc/accumulo:/etc/accumulo:ro,/etc/hadoop:/etc/hadoop:ro,/etc/hbase:/etc/hbase:ro
spark.mesos.executor.gpus	0
spark.mesos.executor.home	/usr/local/spark
spark.mesos.executor.secret.envkeys	(ukryte)
spark.mesos.executor.secret.values	(ukryte)
spark.mesos.principal	kchoromanski
spark.mesos.role	kchoromanski
spark.mesos.secret	(ukryte)
spark.network.timeout	800
spark.rdd.compress	True
spark.repl.local.jars	file:///usr/local/cenagis/jars/geomesa-accumulo-spark-runtime-accumulo1_2.11-3.1.2-

Nazwa	Wartość
	cenagis.jar,file:///usr/local/cenagis/jars/geomesa-cqengine-datastore_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/geomesa-cqengine_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/geomesa-fs-spark-runtime_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/geomesa-fs-spark_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/geomesa-spark-converter_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/geomesa-z3_2.11-3.1.2-cenagis.jar,file:///usr/local/cenagis/jars/cenagis-raster-assembly-1.0.jar
spark.scheduler.mode	FAIR
spark.security.credentials.renewalRatio	0.00000000000001
spark.serializer	org.apache.spark.serializer.KryoSerializer
spark.serializer.objectStreamReset	100
spark.shuffle.service.enabled	true
spark.sql.broadcastTimeout	-1
spark.sql.crossJoin.enabled	true
spark.sql.execution.arrow.enabled	true
spark.sql.Parquet.binaryAsString	true
spark.sql.Parquet.blockSize	268435456
spark.sql.Parquet.compression.codec	snappy
spark.sql.repl.eagerEval.enabled	true
spark.sql.repl.eagerEval.truncate	100
spark.sql.shuffle.partitions	32
spark.submit.deployMode	client
spark.ui.allowFramingFrom	*
spark.ui.showConsoleProgress	true
spark.yarn.dist.files	/usr/local/spark/python/lib/py4j-0.10.7-src.zip,/usr/local/spark/python/lib/pyspark.zip,/tmp/spark-python-3.7/geomesa_pyspark-

Nazwa	Wartość
	3.1.1.dev0+cenagis1.0.zip,/tmp/spark-python-3.7/pytz-2022.7.1.zip,/tmp/spark-python-3.7/pyrasterframes-0.9.1.zip,/tmp/spark-python-3.7/geopyspark.zip
spark.yarn.keytab	(ukryte)
spark.yarn.principal	kchoromanski

Właściwości systemu

Tabela 37. Zestawienia informacji o właściwościach systemu wykorzystywanych do przeprowadzenia badań (opracowanie własne)

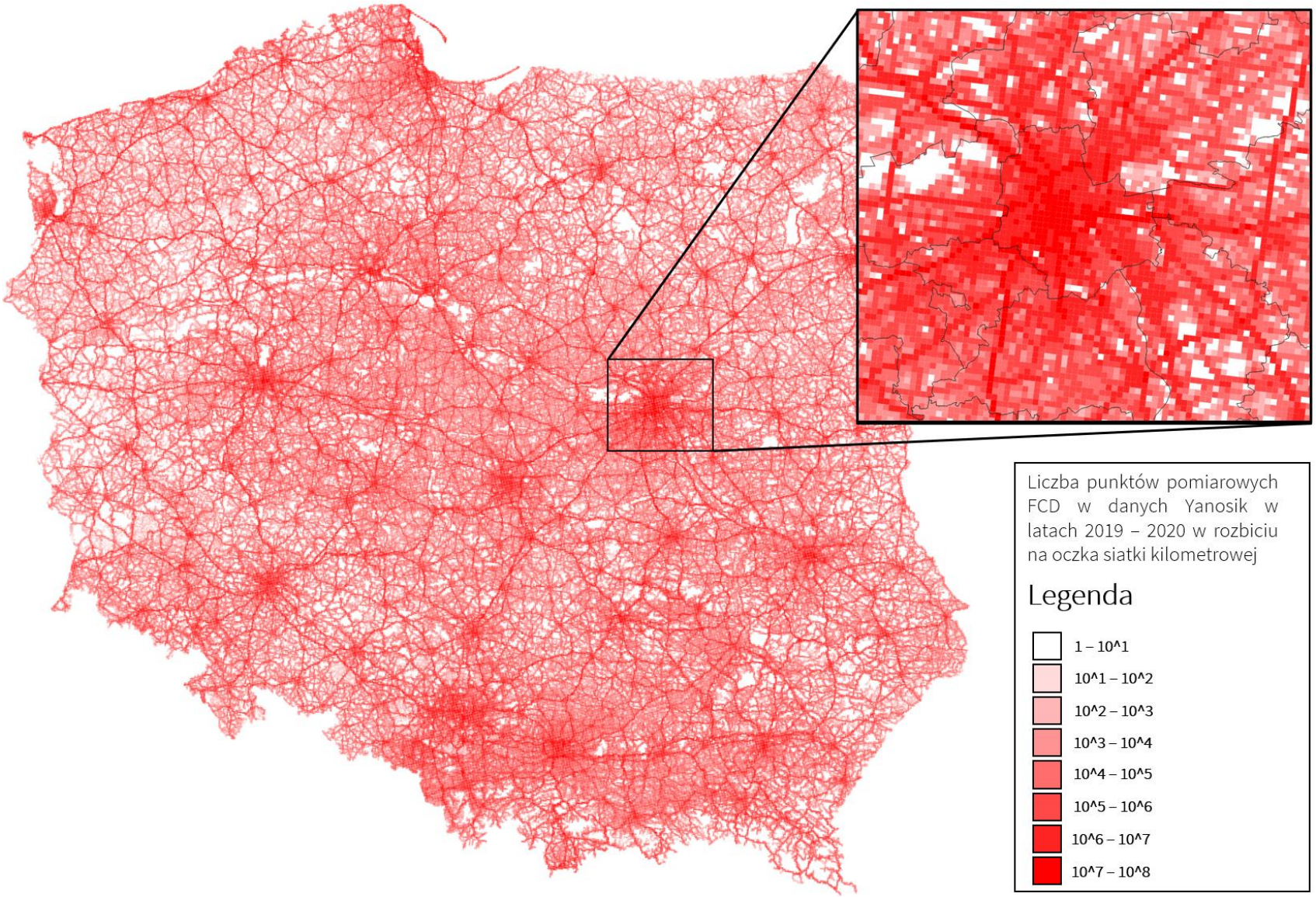
Nazwa	Wartość
SPARK_SUBMIT	true
awt.toolkit	sun.awt.X11.XToolkit
file.encoding	UTF-8
file.encoding.pkg	sun.io
file.separator	/
java.awt.graphicsenv	sun.awt.X11GraphicsEnvironment
java.awt.printerjob	sun.print.PSPrinterJob
java.class.version	52.0
java.endorsed.dirs	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/endorsed
java.ext.dirs	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/ext:/usr/java/packages/lib/ext
java.home	/usr/lib/jvm/java-8-openjdk-amd64/jre
java.io.tmpdir	/tmp
java.library.path	/usr/local/nvidia/lib64:/usr/local/nvidia/lib64:/usr/local/lib:/usr/local/lib/x86_64-linux-gnu:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-gnu:/usr/java/packages/lib/amd64:/usr/lib/x86_64-linux-gnu/jni:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-gnu:/usr/lib/jni:/lib:/usr/lib
java.runtime.name	OpenJDK Runtime Environment
java.runtime.version	1.8.0_352-8u352-ga-1~18.04-b08

Nazwa	Wartość
java.specification.maintenance.version	4
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.8
java.vendor	Private Build
java.vendor.url	http://java.oracle.com/
java.vendor.url.bug	http://bugreport.sun.com/bugreport/
java.version	1.8.0_352
java.vm.info	mixed mode
java.vm.name	OpenJDK 64-Bit Server VM
java.vm.specification.name	Java Virtual Machine Specification
java.vm.specification.vendor	Oracle Corporation
java.vm.specification.version	1.8
java.vm.vendor	Private Build
java.vm.version	25.352-b08
javax.accessibility.assistive_technologies	org.GNOME.Accessibility.AtkWrapper
jetty.git.hash	unknown
line.separator	
os.arch	amd64
os.name	Linux
os.version	3.10.0-1160.62.1.el7.x86_64
path.separator	:
sun.arch.data.model	64
sun.boot.class.path	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/resources.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/rt.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/sunrsasign.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jsse.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jce.jar:/usr/lib/jvm/java-8-openjdk-

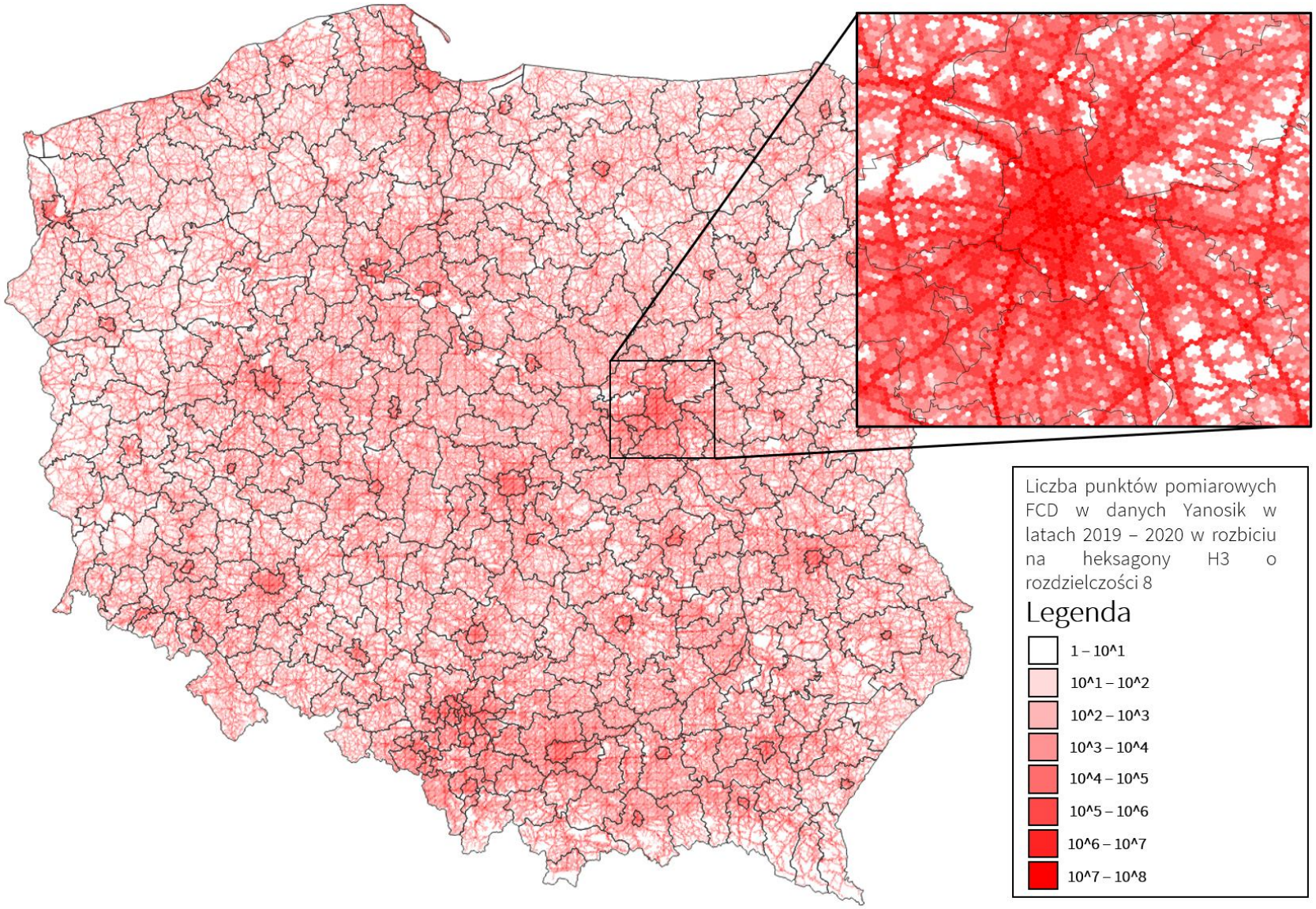
Nazwa	Wartość
	amd64/jre/lib/charsets.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jfr.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/classes
sun.boot.library.path	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/amd64
sun.cpu.endian	little
sun.cpu.isalist	
sun.font.fontmanager	sun.awt.X11FontManager
sun.io.unicode.encoding	UnicodeLittle
sun.java.command	*********(redacted)
sun.java.launcher	SUN_STANDARD
sun.jnu.encoding	UTF-8
sun.management.compiler	HotSpot 64-Bit Tiered Compilers
sun.nio.ch.bugLevel	
sun.os.patch.level	unknown
user.dir	/home/CENAGIS/kchoromanski/mobility/notebooks/segments_analysis
user.home	/home/CENAGIS/kchoromanski
user.language	en
user.name	kchoromanski
user.timezone	Etc/UTC

Załącznik 2 – wyniki scenariusza B2

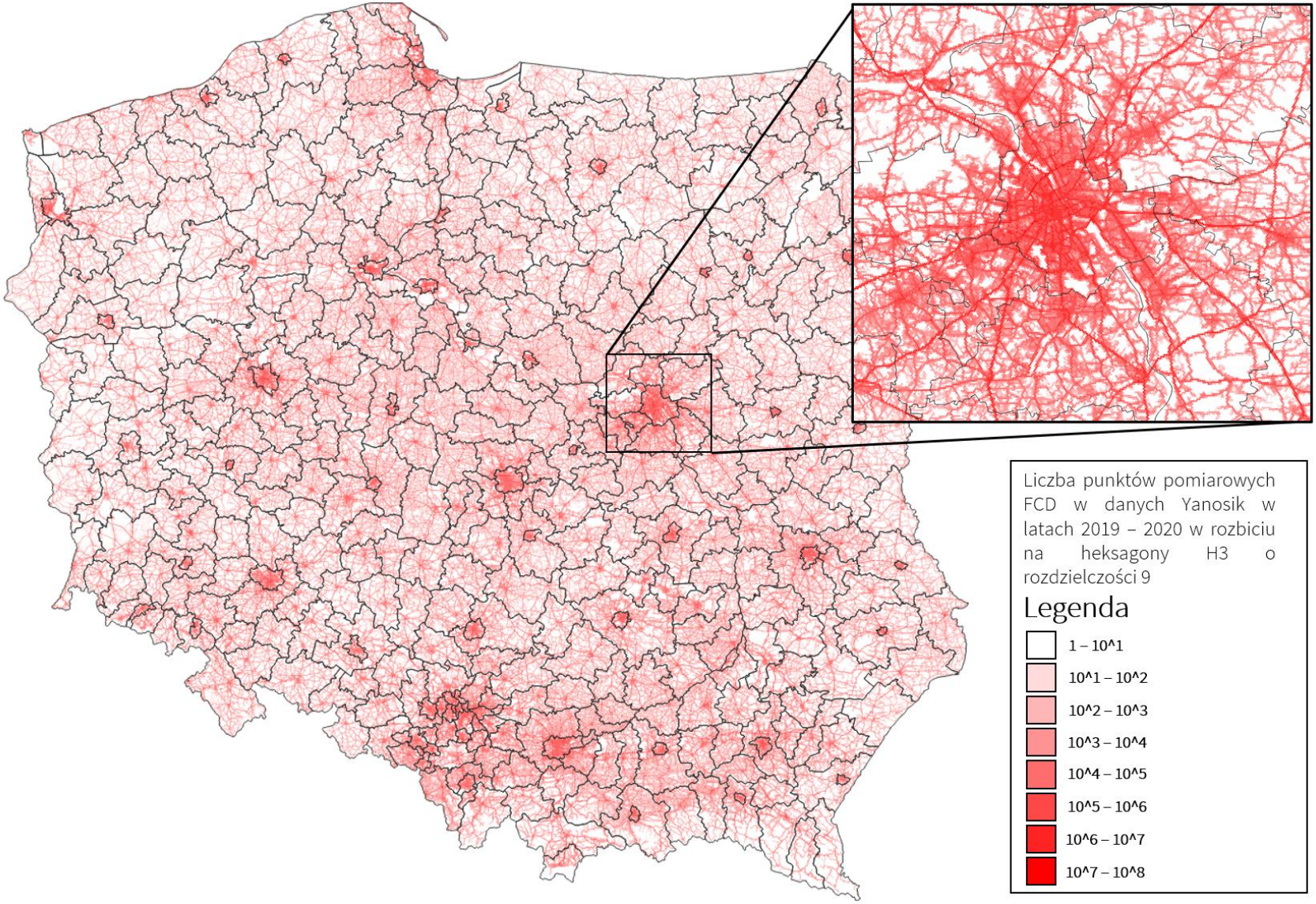
Na zaprezentowany poniżej zestawie wizualizacji kartograficznych przedstawiono przestrzenny rozkład punktów pomiarowych FCD pozyskanych z aplikacji Yanosik w latach 2019 - 2020, zagregowanych do badanych jednostek przestrzennych: siatki kilometrowej, heksagonów h3 o dwóch różnych rozdzielczościach, obwodów spisowych oraz gmin. Niezależnie od zastosowanej jednostki agregacji zaobserwować można wspólne cechy przedstawionych wyników. Największe zagęszczenia punktów występują wokół dużych miast oraz wzdłuż głównych arterii takich jak autostrady i drogi ekspresowe. Zastosowanie różnych agregacji pozwala jednak również na rozszerzenie wniosków płynących z analizy danych. Produkty o wyższej rozdzielczości takie jak siatka kilometrowa czy heksagony H3 umożliwiają analizę specyfiki ruchu również wewnątrz miast. W szczególności zastosowanie heksagonów o wyższej rozdzielczości pozwala na obserwacje natężenia na głównych ulicach miast z wysoką dokładnością. Te produkty pozwalają również na identyfikacje głównych połączeń drogowych pomiędzy miastami i ocenę intensywności ich wykorzystania. W przypadku obszarów statystycznych interpretowalność jest niższa. Ich granice nie uwzględniają występowania połączeń drogowych, przez co oryginalne dane zostały uśrednione w sposób nieprzynoszący istotnych korzyści. Taka agregacja umożliwia jednak uwzględnienie danych w analizie razem z innymi rodzajami zbiorów statystycznych. Należy jednak pamiętać, że dane w tym przypadku mogły zostać zniekształcone. Agregacja na poziomie gmin powoduje największą utratę informacji z oryginalnych danych, pozwala ona jednak na łatwe porównanie natężenia ruchu pomiędzy poszczególnymi gminami, co może mieć istotną wartość informacyjną.



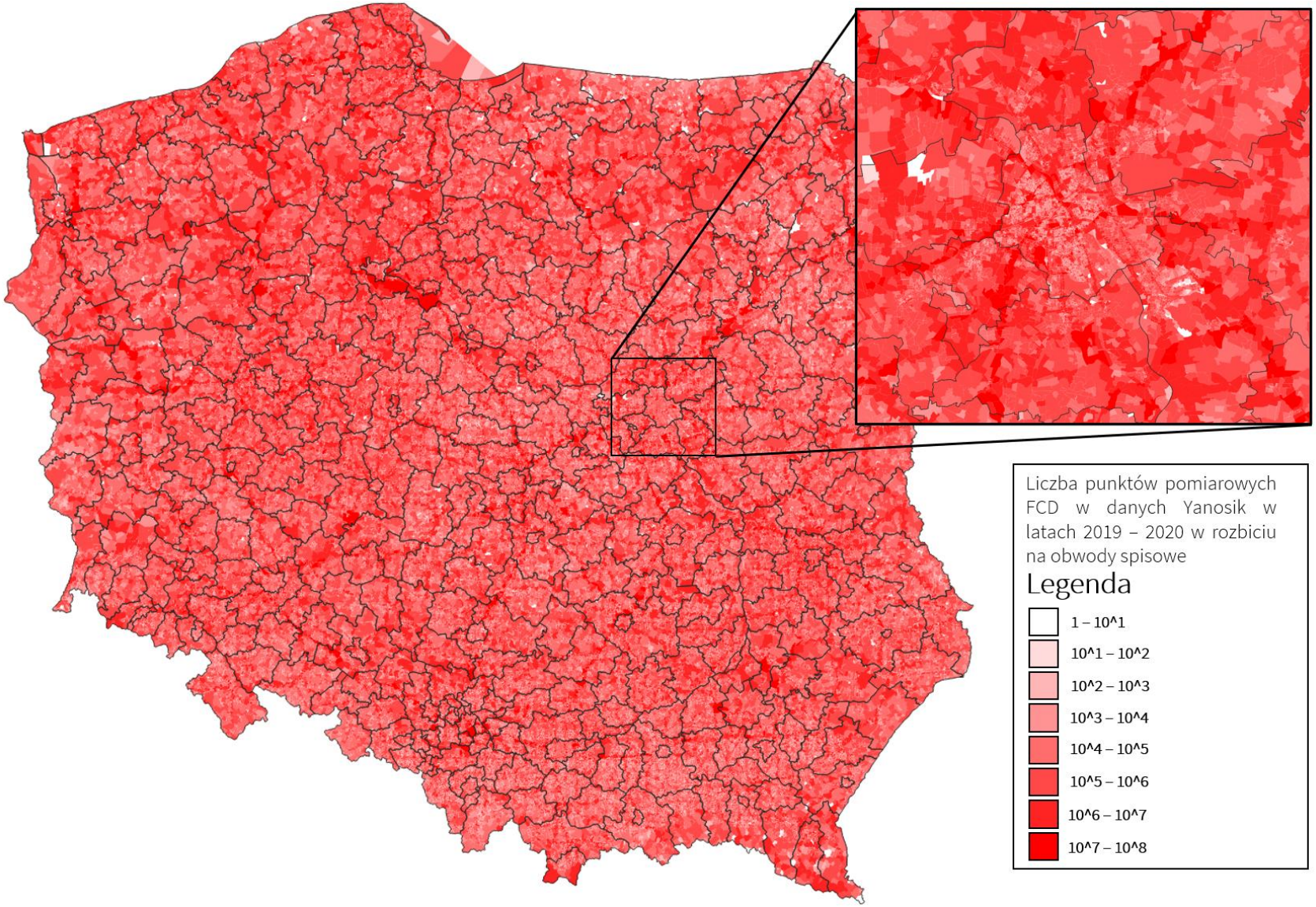
Rysunek 91. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik w oczkach siatki kilometrowej (opracowanie własne)



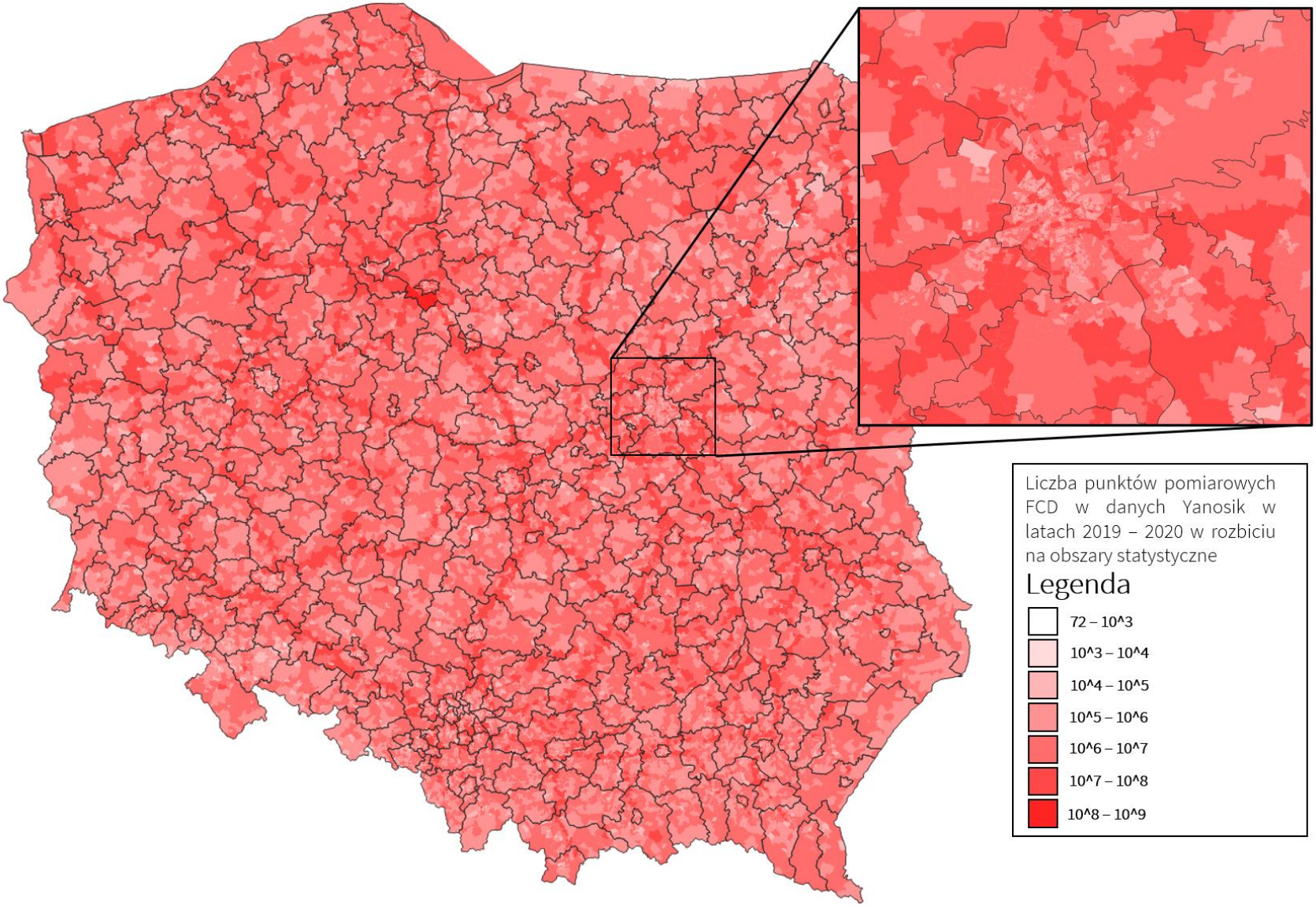
Rysunek 92. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik w heksagonach H3 o rozdzielczości 8 (opracowanie własne)



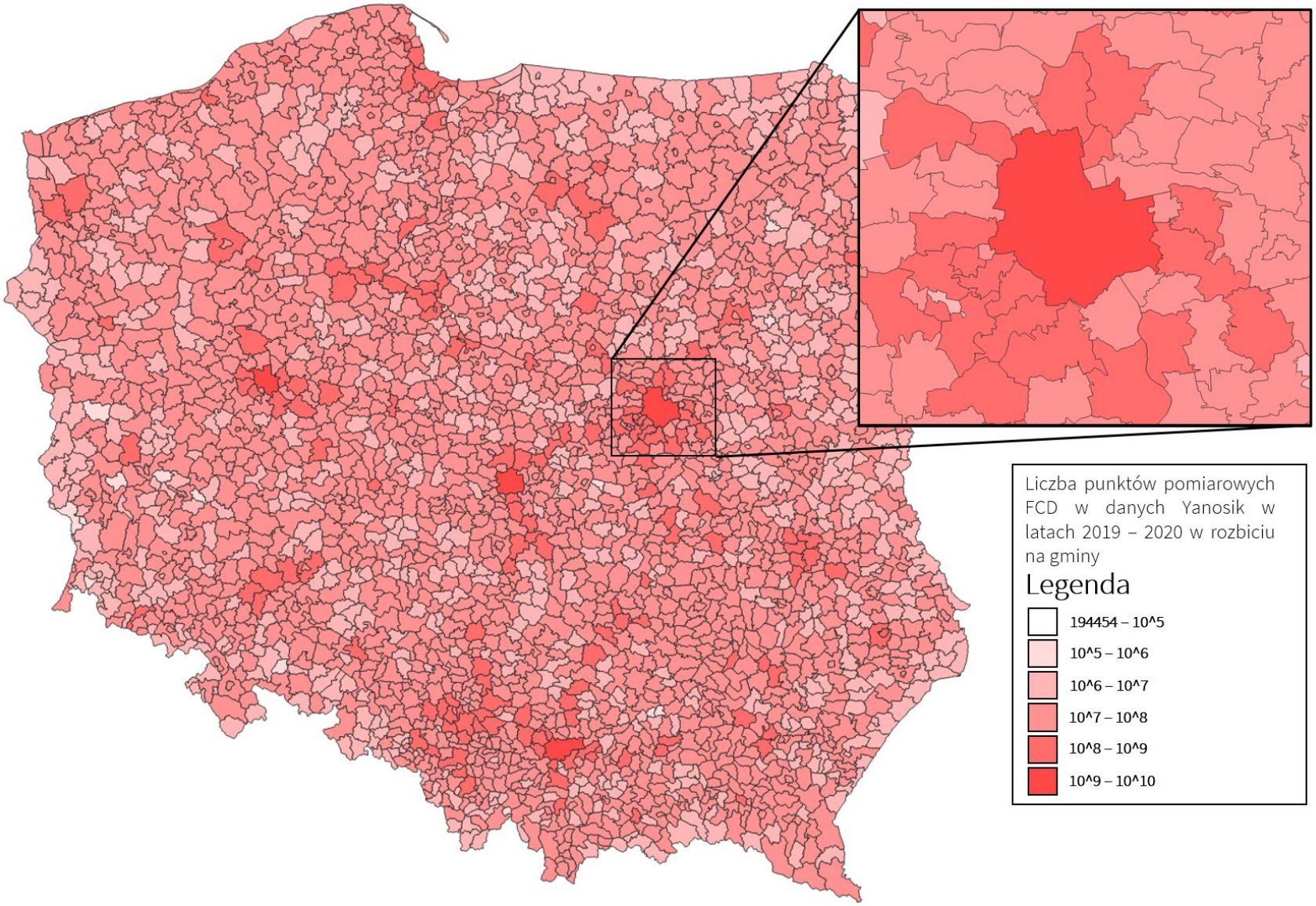
Rysunek 93. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik w heksagonach H3 o rozdzielczości 9 (opracowanie własne)



Rysunek 94. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik obwodach spisowych (opracowanie własne)



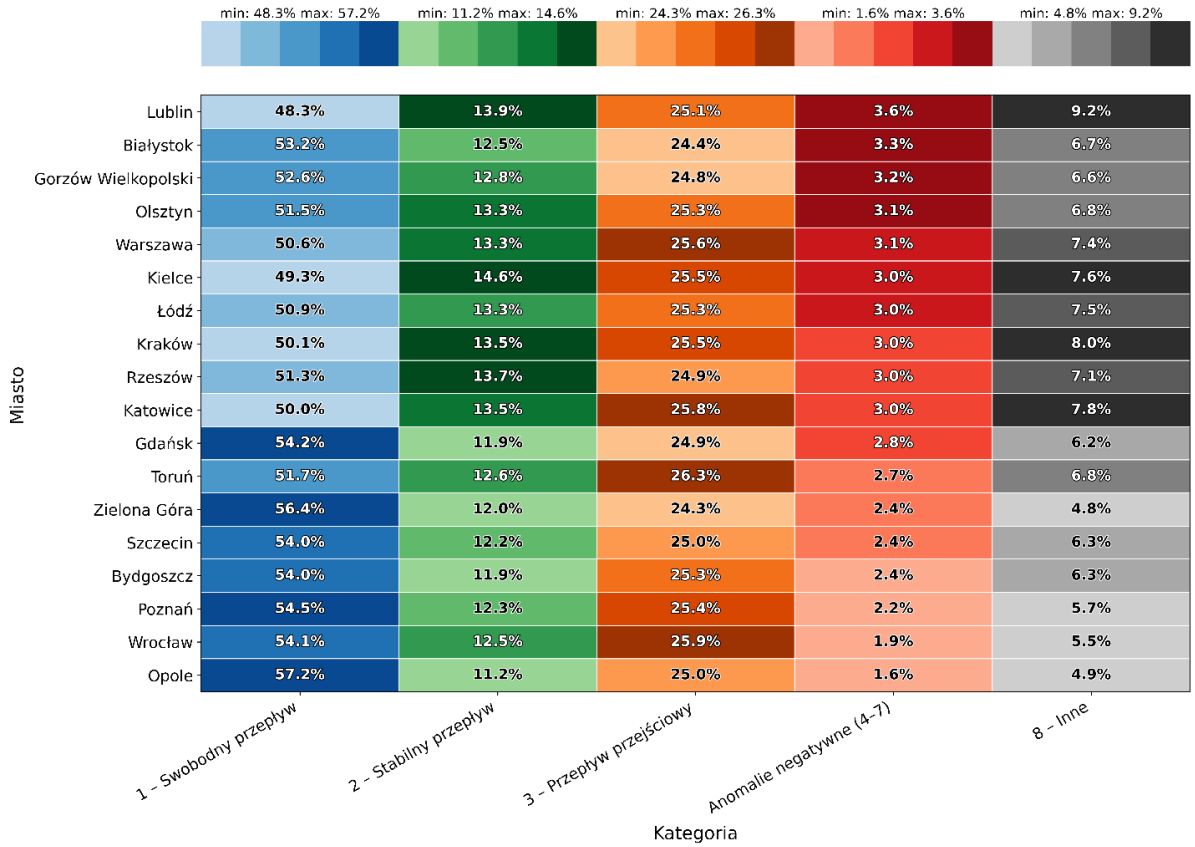
Rysunek 95. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik w obszarach statystycznych (opracowanie własne)



Rysunek 96. Wynik realizacji zliczenia punktów pomiarowych FCD z danych Yanosik w gminach (opracowanie własne)

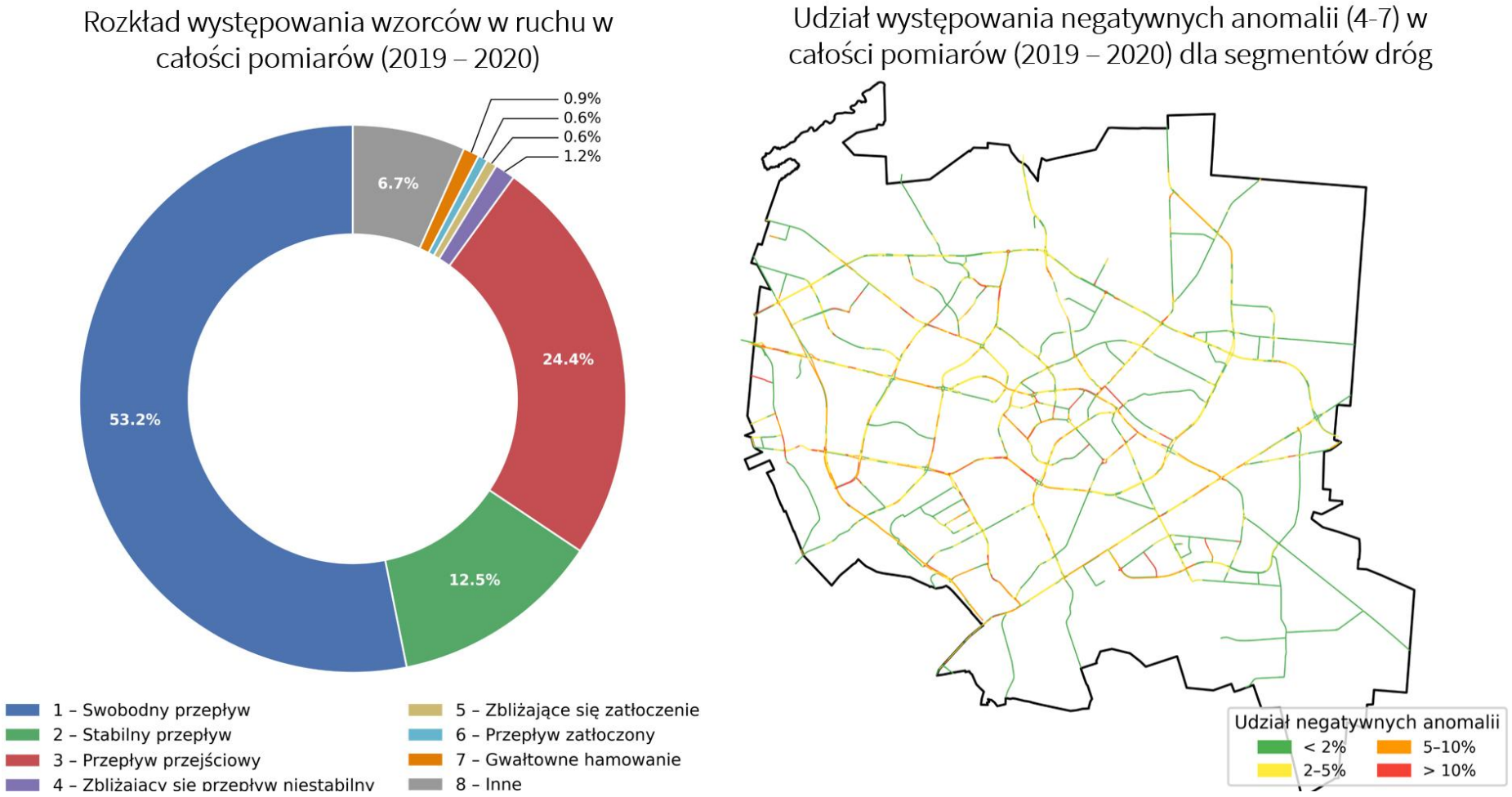
Załącznik 3 – wyniki scenariusza B4

Poniżej przedstawiono wynik analizy przeprowadzonej na podstawie realizacji scenariusza B4. Zagregowany wynik realizacji scenariusza zaprezentowano w podziale na miasta wojewódzkie. Przedstawiono rozkład występowania poszczególnych wzorców w ruchu w postaci wykresu kołowego zestawionego z wizualizacją kartograficzną prezentującą procentowy udział występowania anomalii negatywnych (zbliżający się przepływ niestabilny, zbliżające się zatłoczenie, przepływ zatłoczony oraz gwałtowne hamowanie) w pomiarach dla poszczególnych segmentów dróg. Takie opracowania mogą stanowić jedno ze źródeł procesu analizy miejskiego ruchu drogowego. Należy jednak pamiętać, że nie są one wolne od błędów, a przed ich wykorzystaniem w procesie decyzyjnym należałoby dokonać walidacji wyników w skali lokalnej (np. z wykorzystaniem pomiarów przy pomocy kamer bądź pętli indukcyjnych) na wybranych segmentach dróg w celu walidacji wyników. Takie opracowanie pozwala jednak na analizę ruchu drogowego w skali całego miasta (bądź innego wybranego obszaru) w celu identyfikacji obszarów zainteresowania do dalszych szczegółowych analiz. Na rysunku 97 przedstawiono zestawienie procentowego udziału poszczególnych wzorców ruchu dla poszczególnych miast. Takie przedstawienie wyników pozwala z kolei na porównanie rozkładu wzorców pomiędzy miastami i analizę w których miastach udział wzorców negatywnych jest większy. Rozkłada się on od 1.6% dla Opola do 3.6% dla Lublina. Biorąc po uwagę, że analizie poddano cały okres pomiarowy dla wszystkich godzin w dobie (w tym tych, w których ruch naturalne będzie niewielki – tj. godziny nocne), różnice te mogą realnie przekładać się na jakość ruchu w miastach. Wnioski te należy jednak również poddać walidacji, gdyż dane wejściowe pochodzące z aplikacji mobilnych mogą cechować się błędami. Co więcej znaczny wpływ na wynik może mieć też układ granic samego miasta (np. w granice danego miasta może wchodzić więcej dróg o małym natężeniu ruchu co wpływa na procentowy rozkład anomalii).



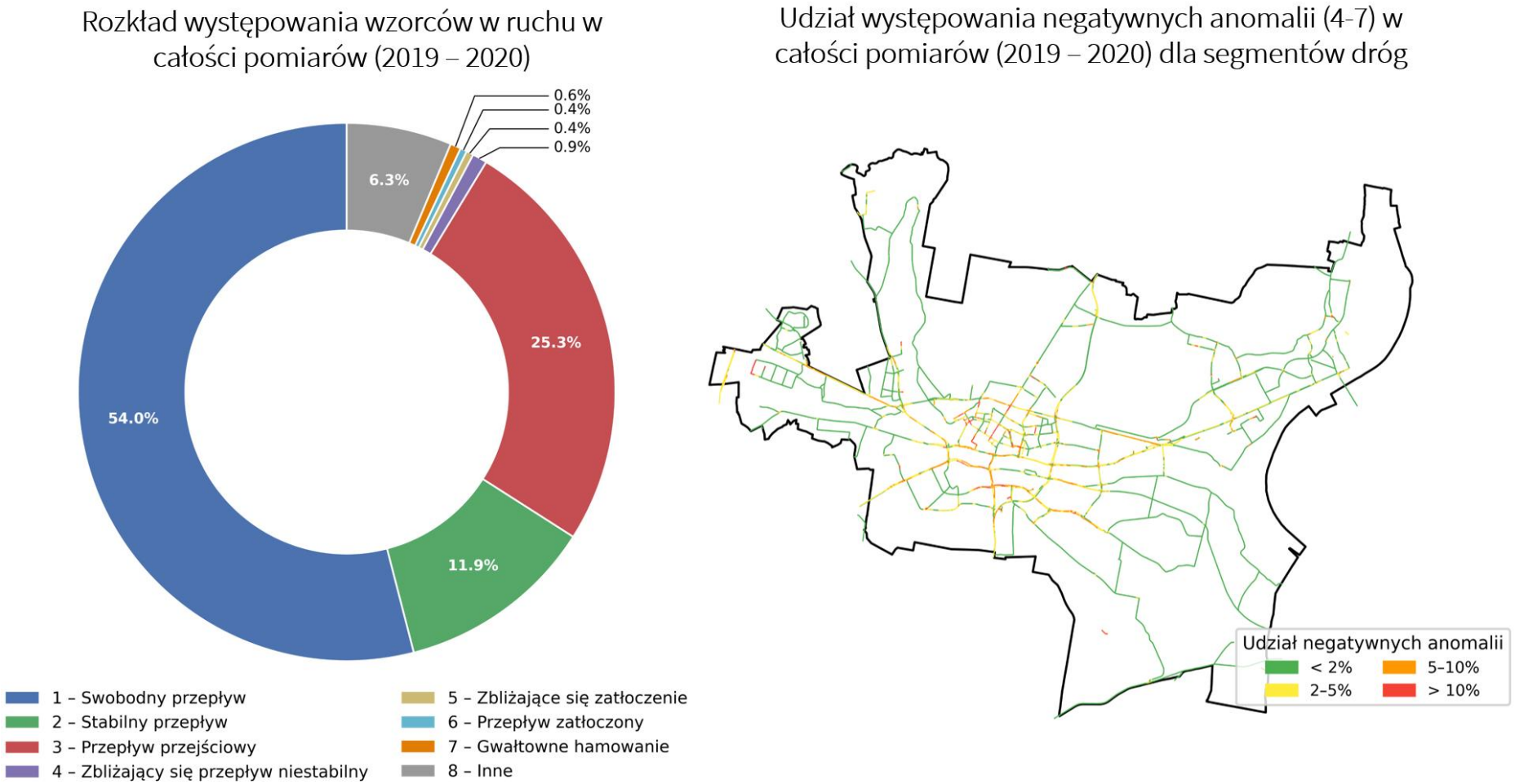
Rysunek 97. Zestawienie występowania poszczególnych wzorców w ruchu drogowym dla miast wojewódzkich wytworzone na podstawie rezultatów scenariusza B4 (opracowanie własne)

Białystok



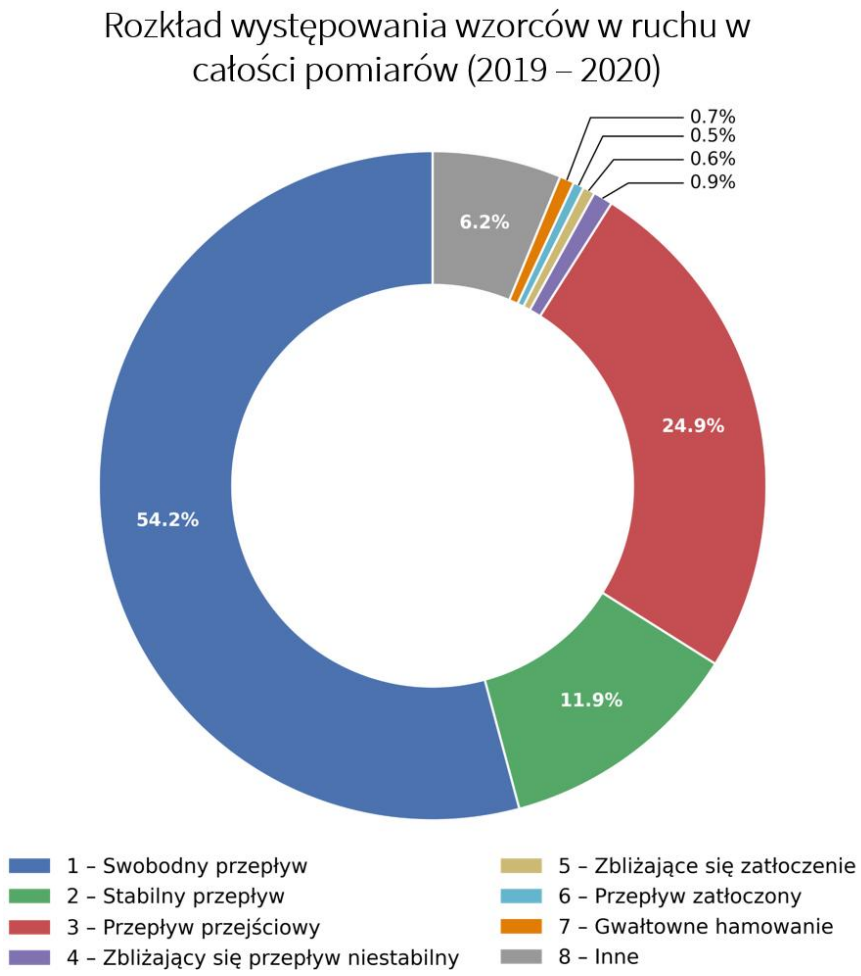
Rysunek 98. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Białegostoku (opracowanie własne1)

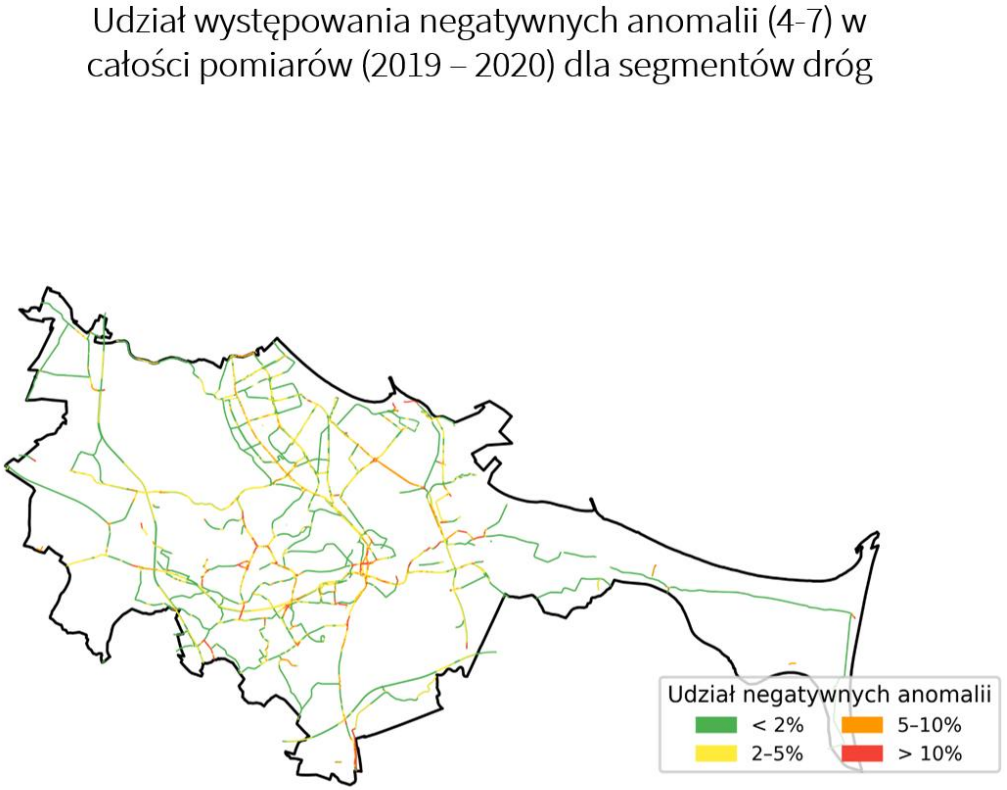
Bydgoszcz



Rysunek 99. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Bydgoszczy (opracowanie własne)

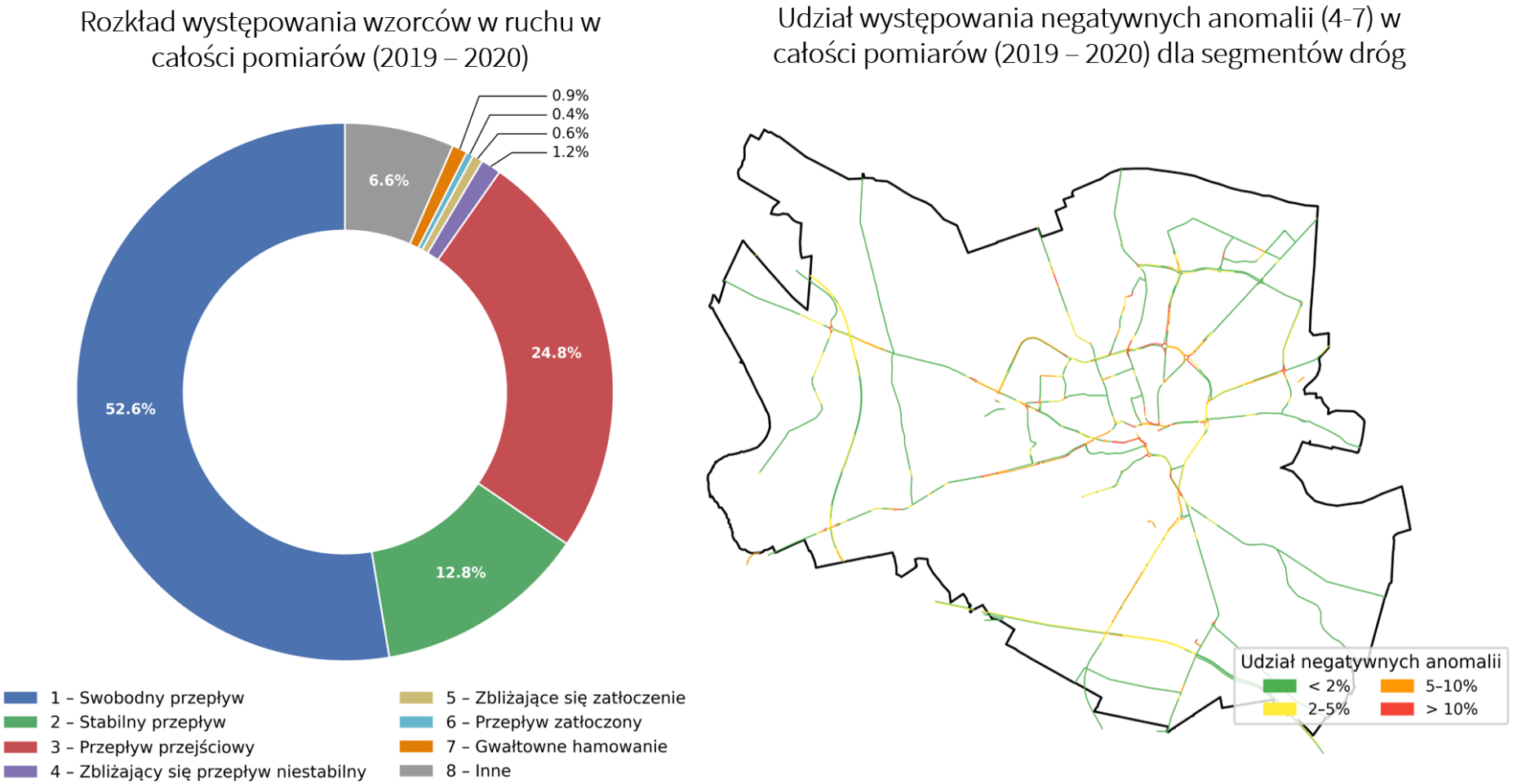
Gdańsk





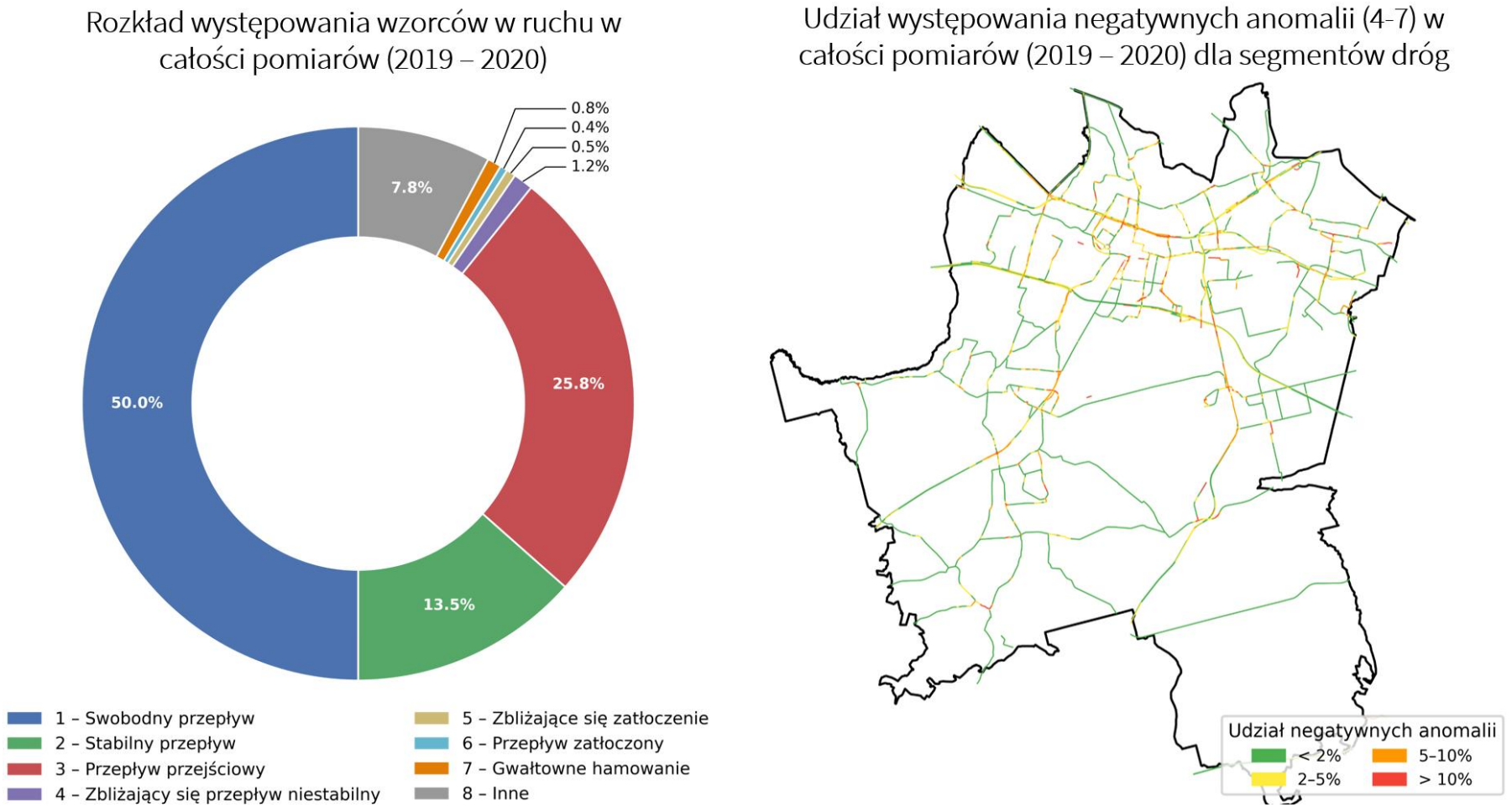
Rysunek 100. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Gdańska (opracowanie własne)

Gorzów Wielkopolski



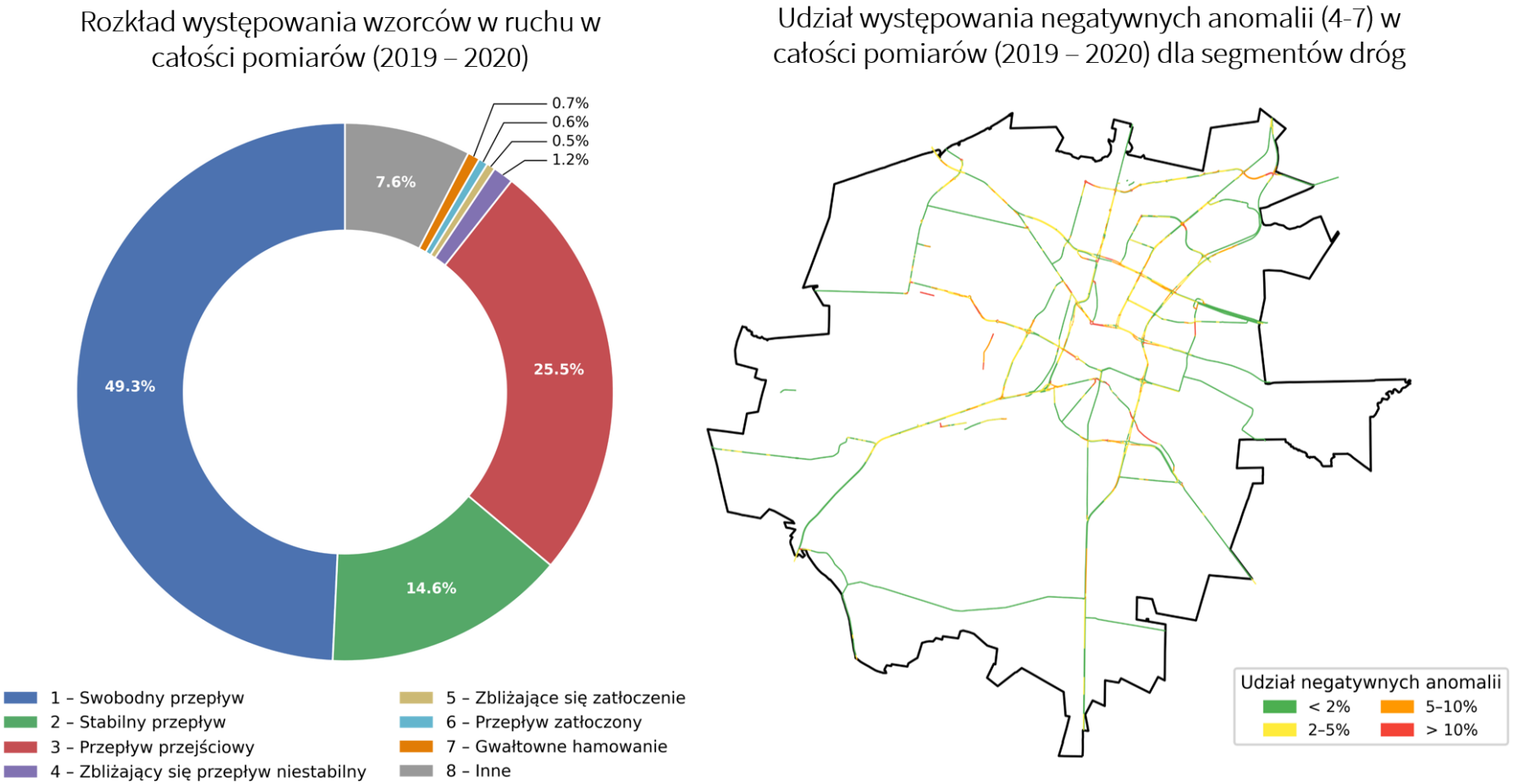
Rysunek 101. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Gorzowa Wielkopolskiego (opracowanie własne)

Katowice



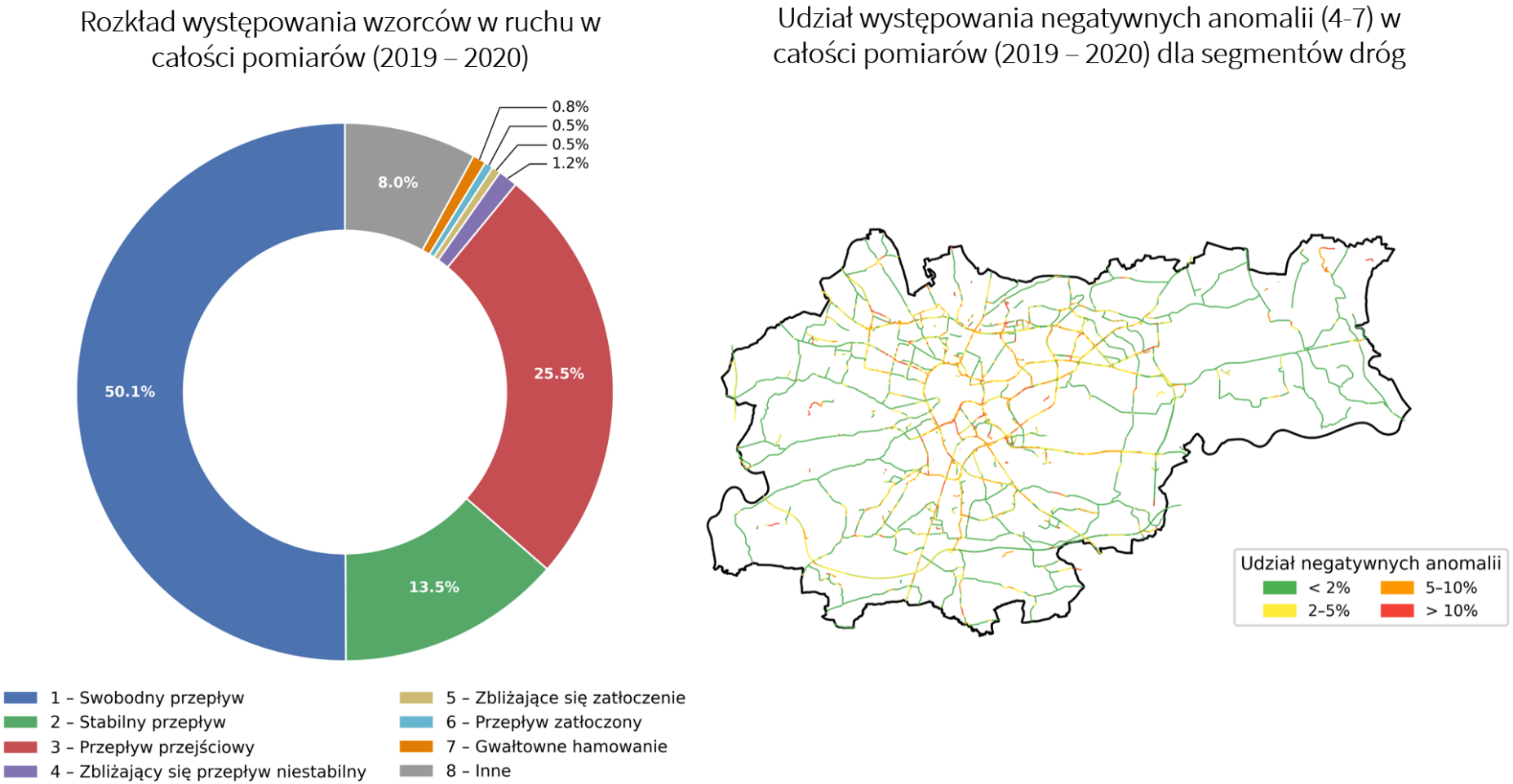
Rysunek 102. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Katowic (opracowanie własne)

Kielce



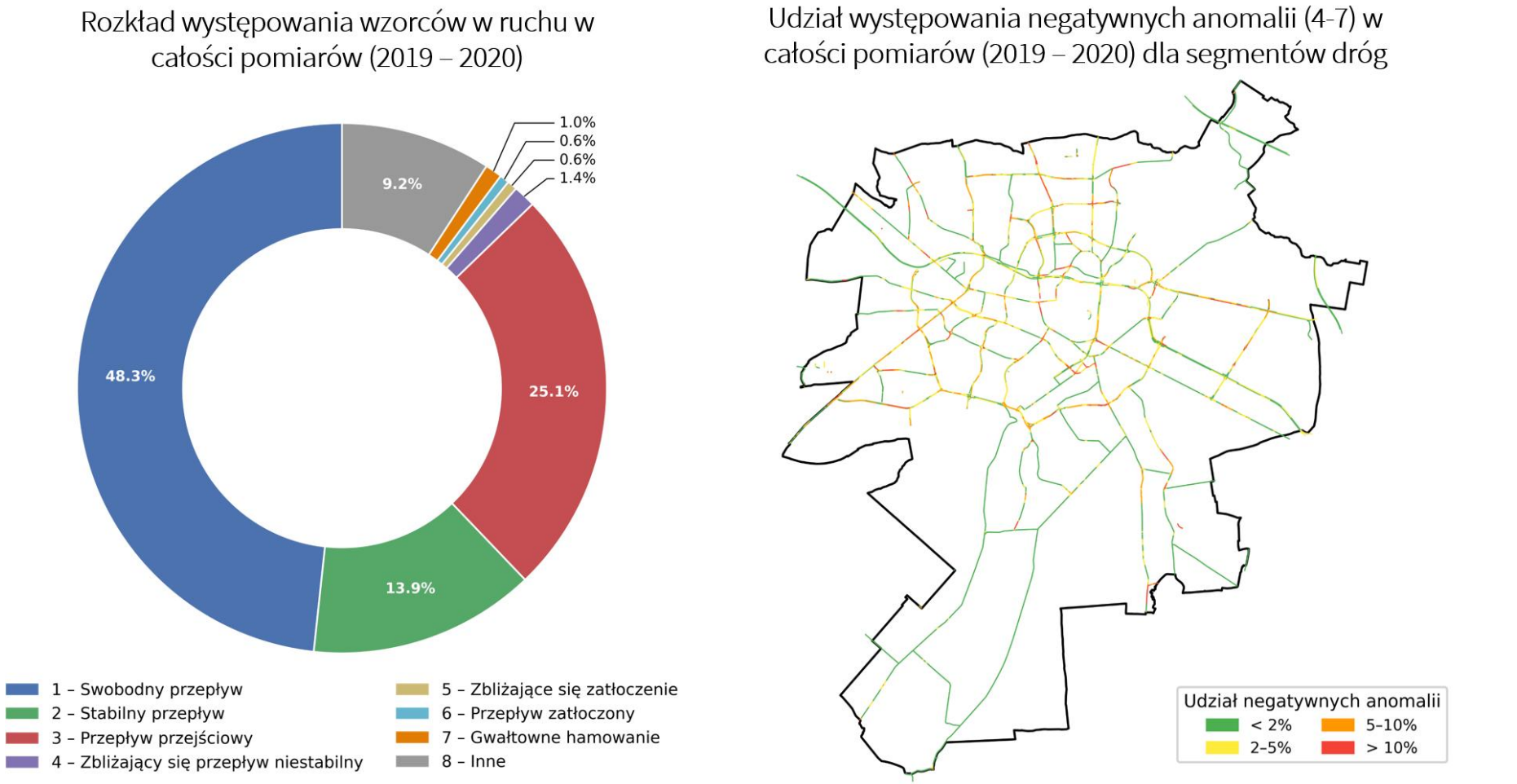
Rysunek 103. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Kielc (opracowanie własne)

Kraków



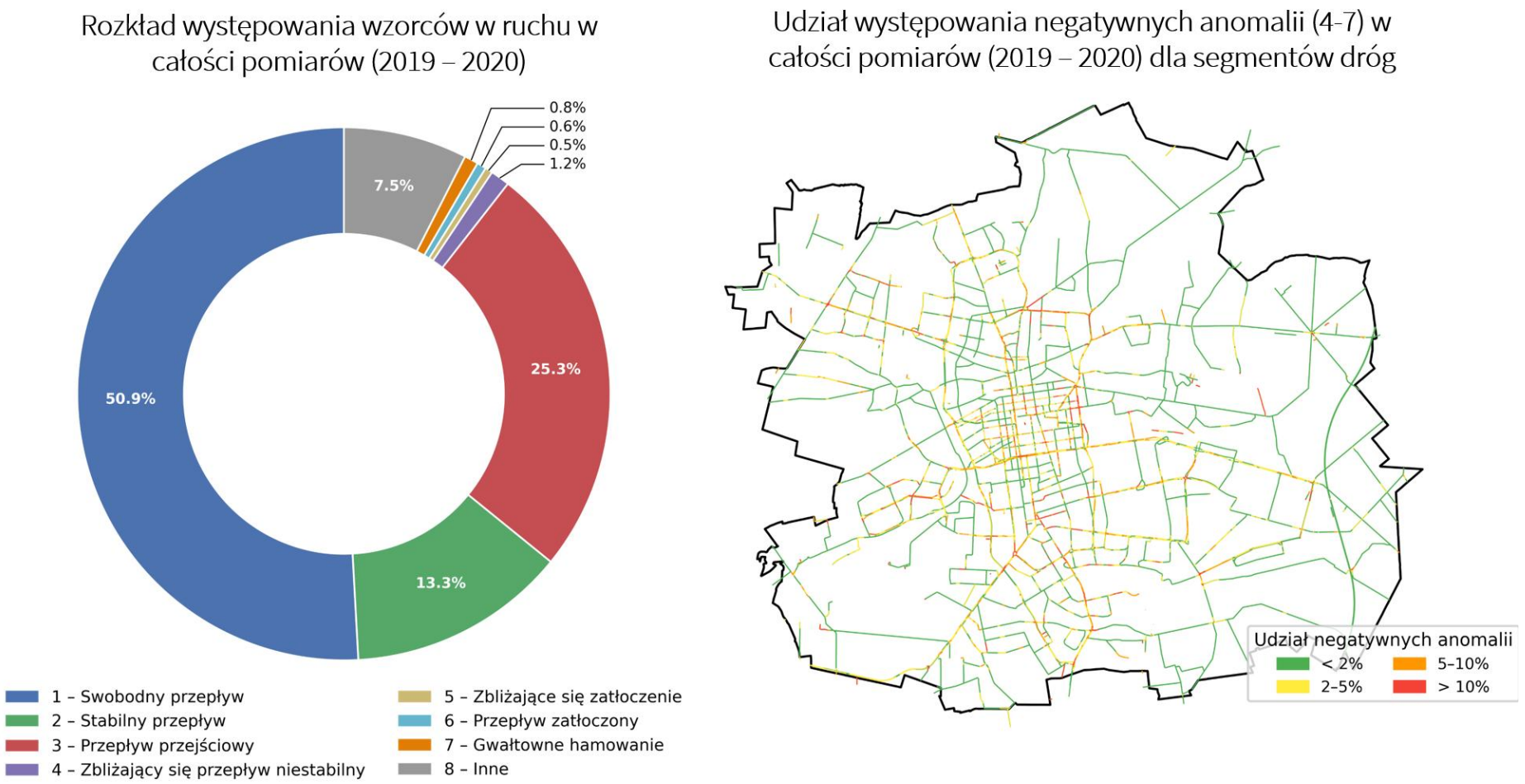
Rysunek 104. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Krakowa (opracowanie własne)

Lublin



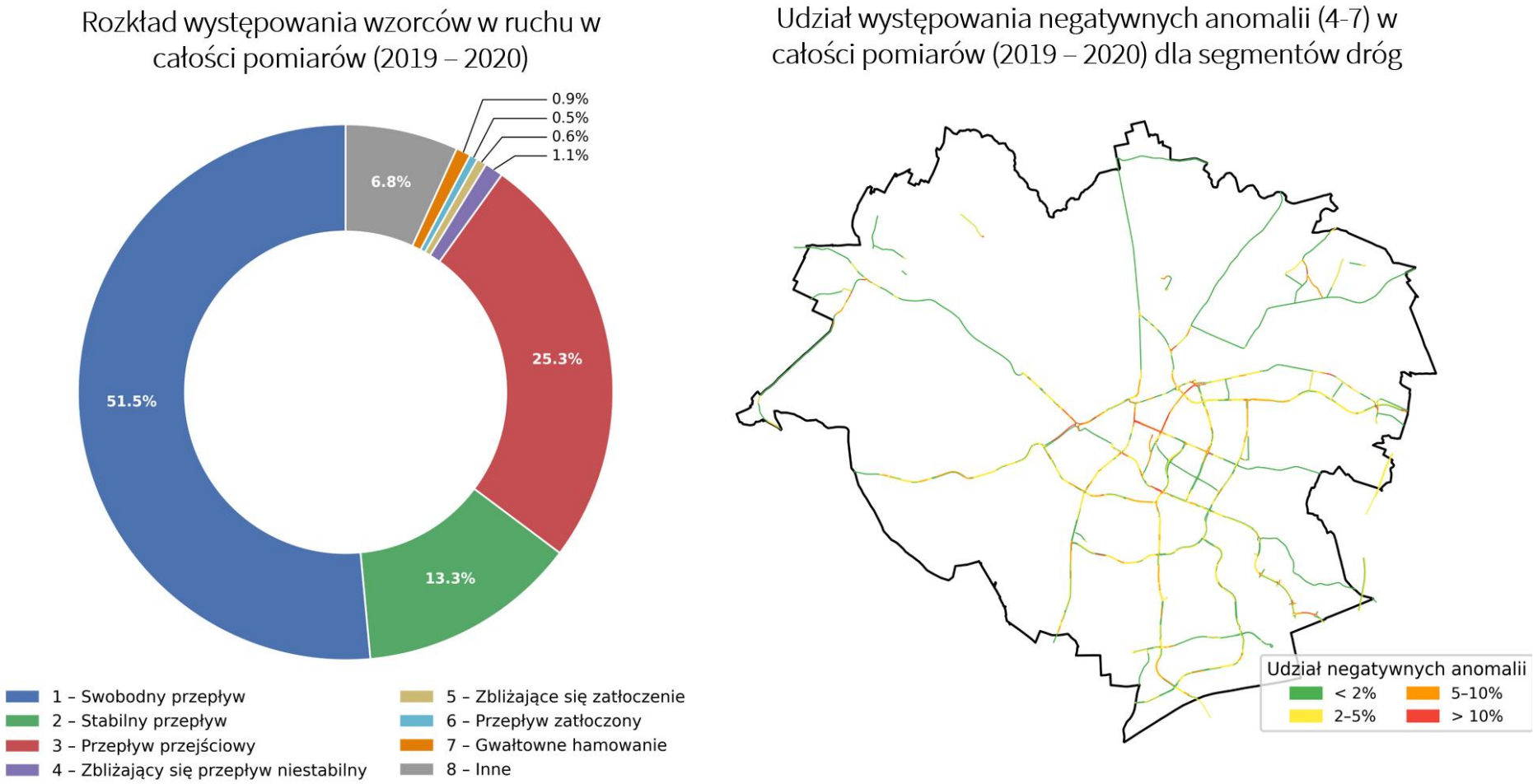
Rysunek 105. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Lublina (opracowanie własne)

Łódź



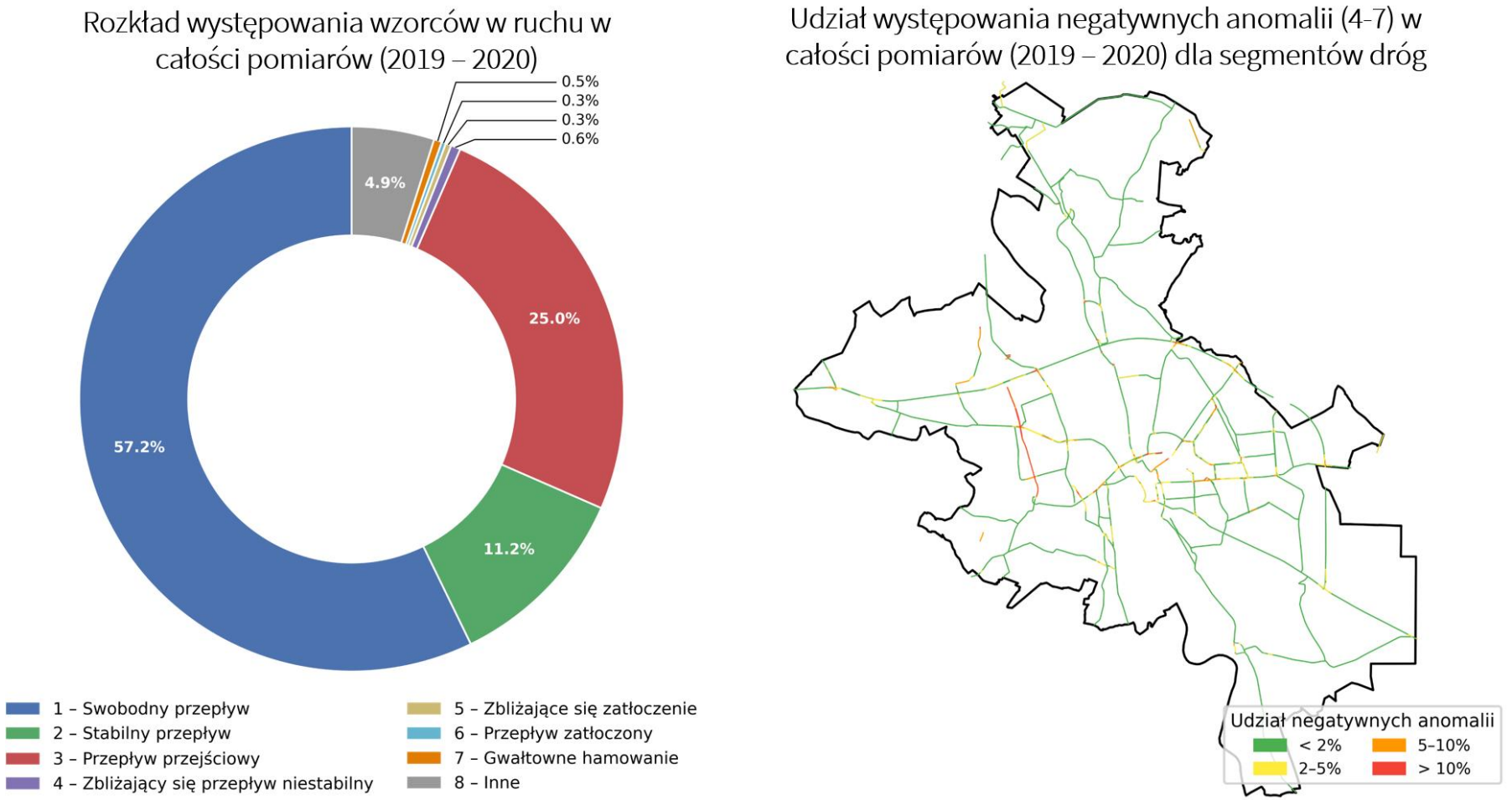
Rysunek 106. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Łodzi (opracowanie własne)

Olsztyn



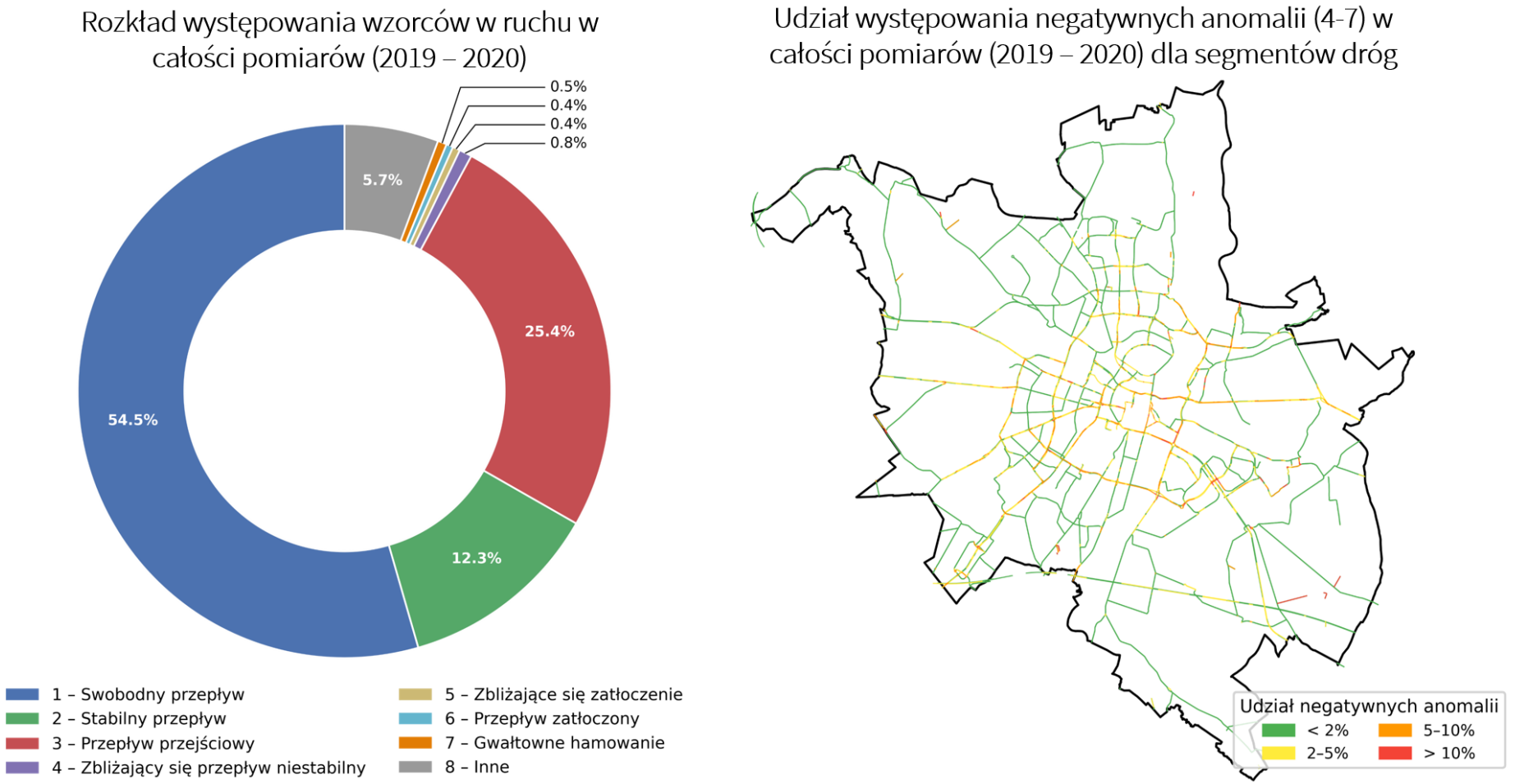
Rysunek 107. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Olsztyna (opracowanie własne)

Opole



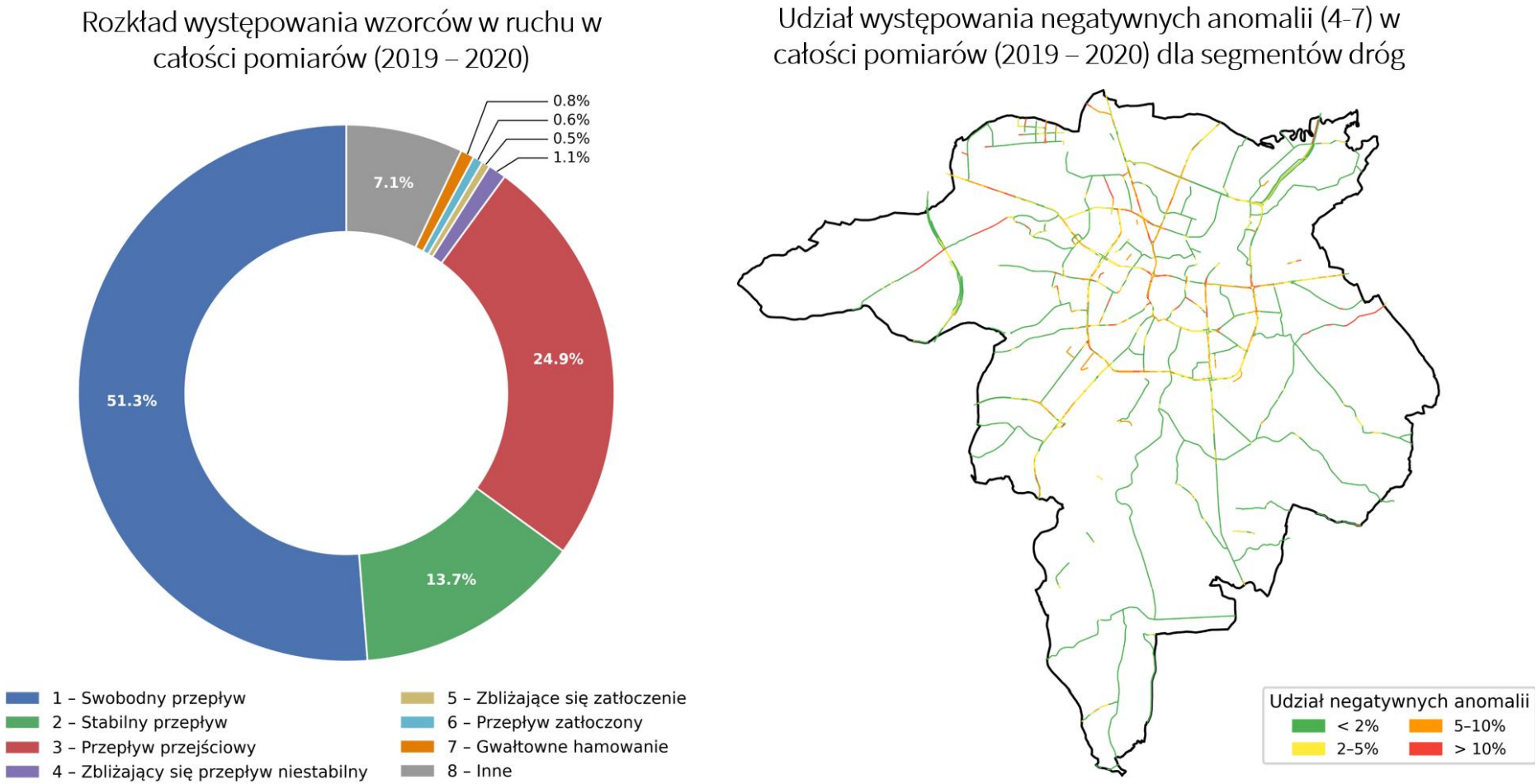
Rysunek 108. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Opola(opracowanie własne)

Poznań



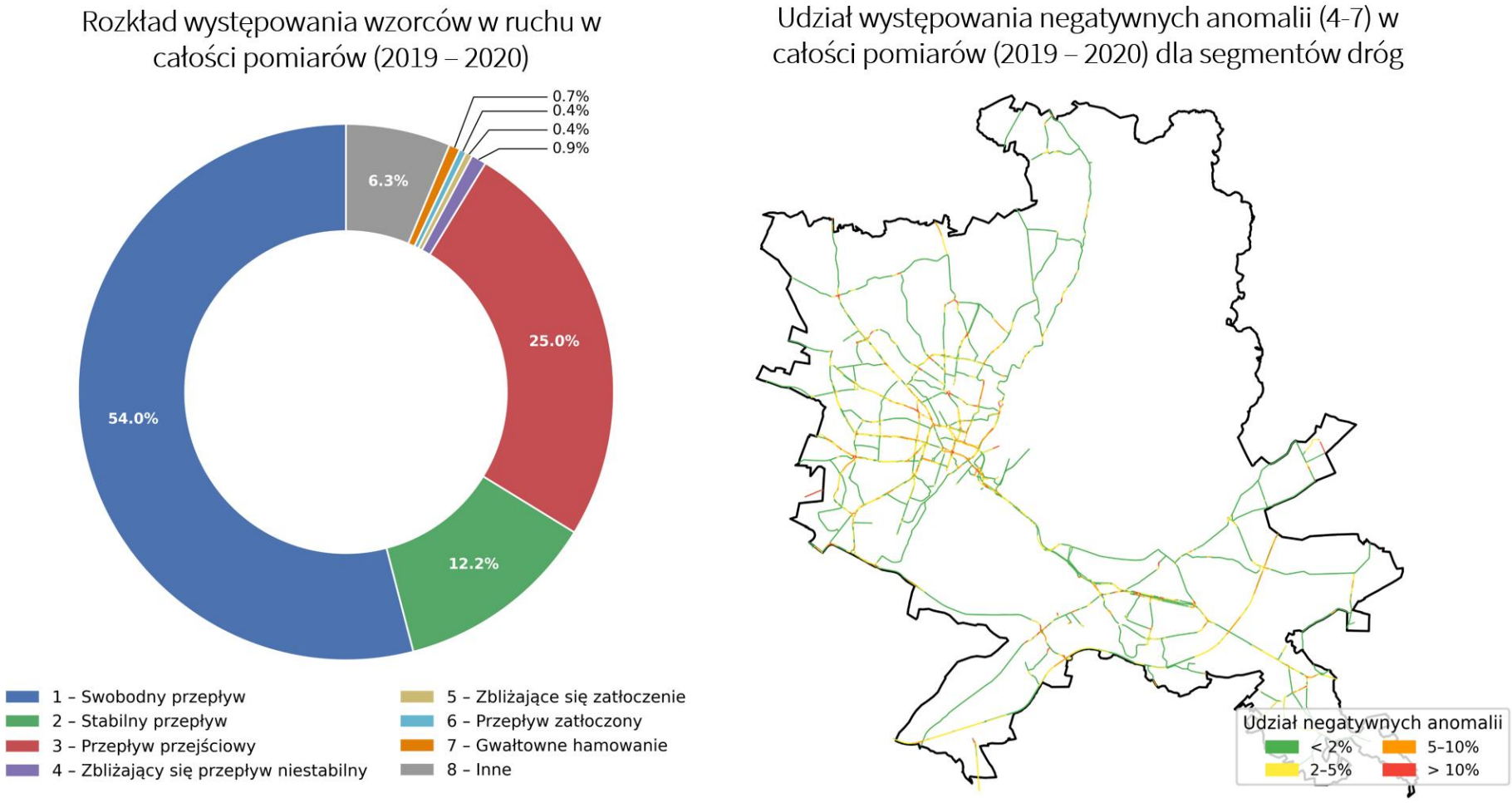
Rysunek 109. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Poznania (opracowanie własne)

Rzeszów



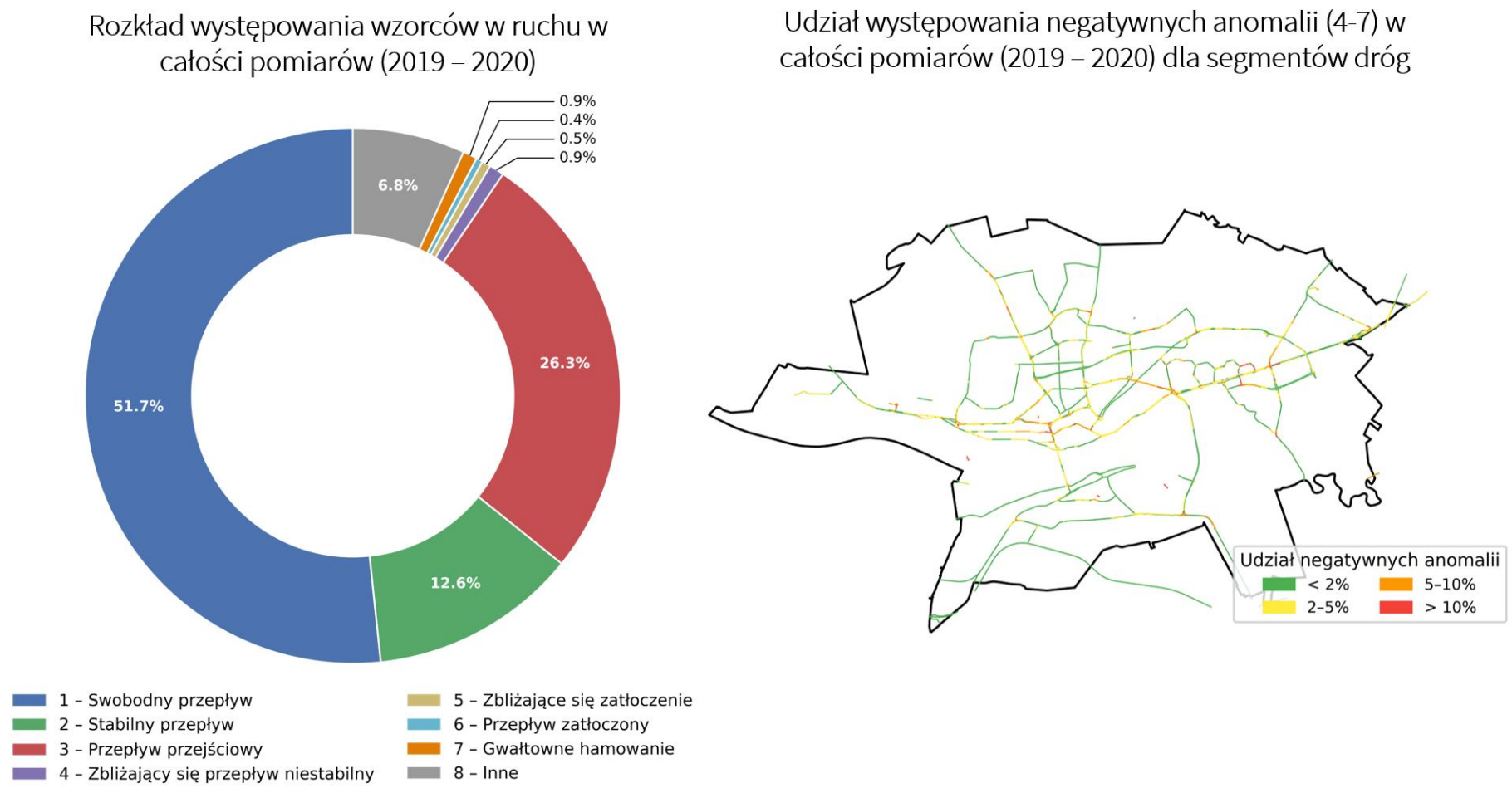
Rysunek 110. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Rzeszowa (opracowanie własne)

Szczecin



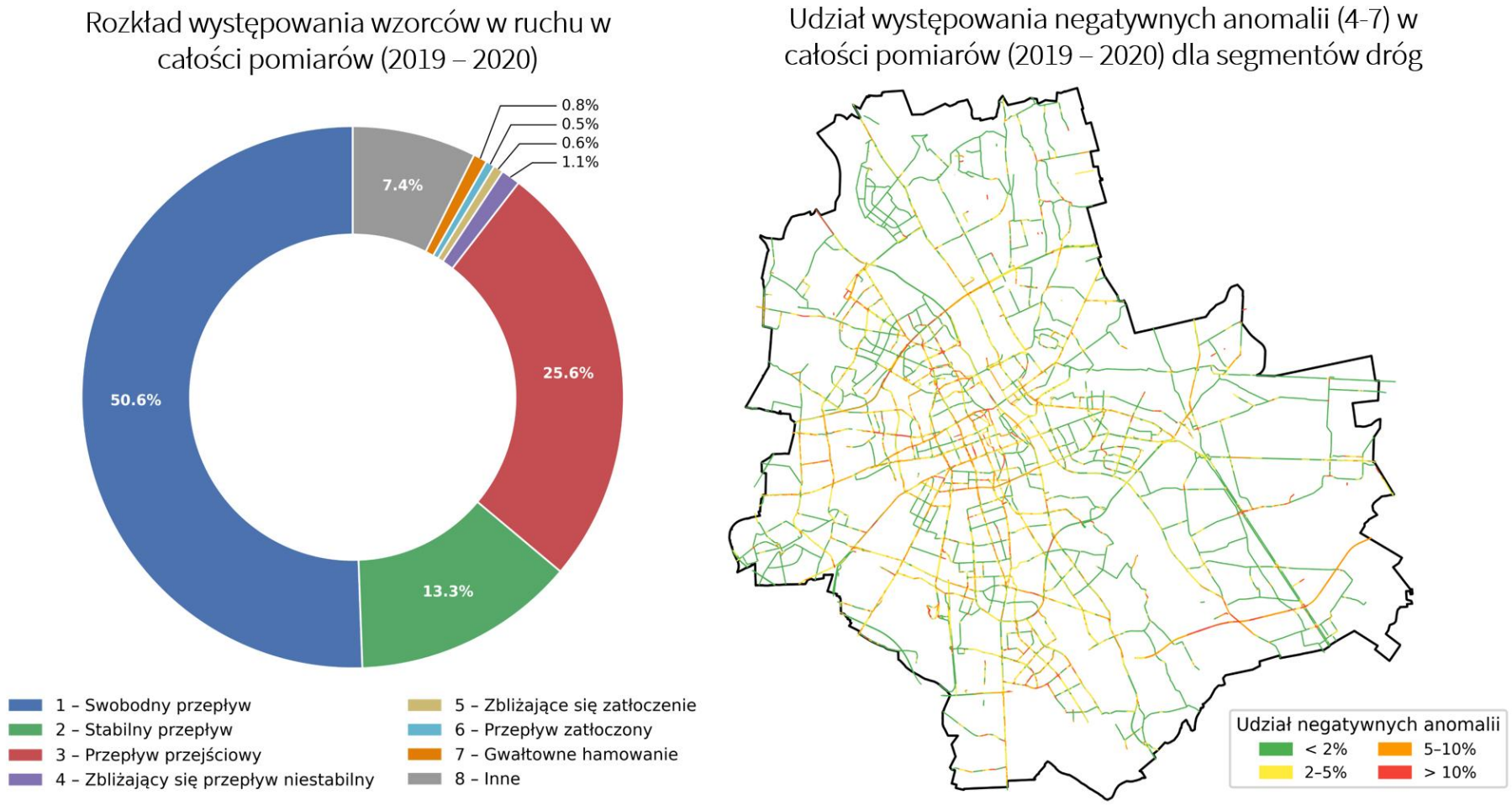
Rysunek 111. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Szczecina (opracowanie własne)

Toruń



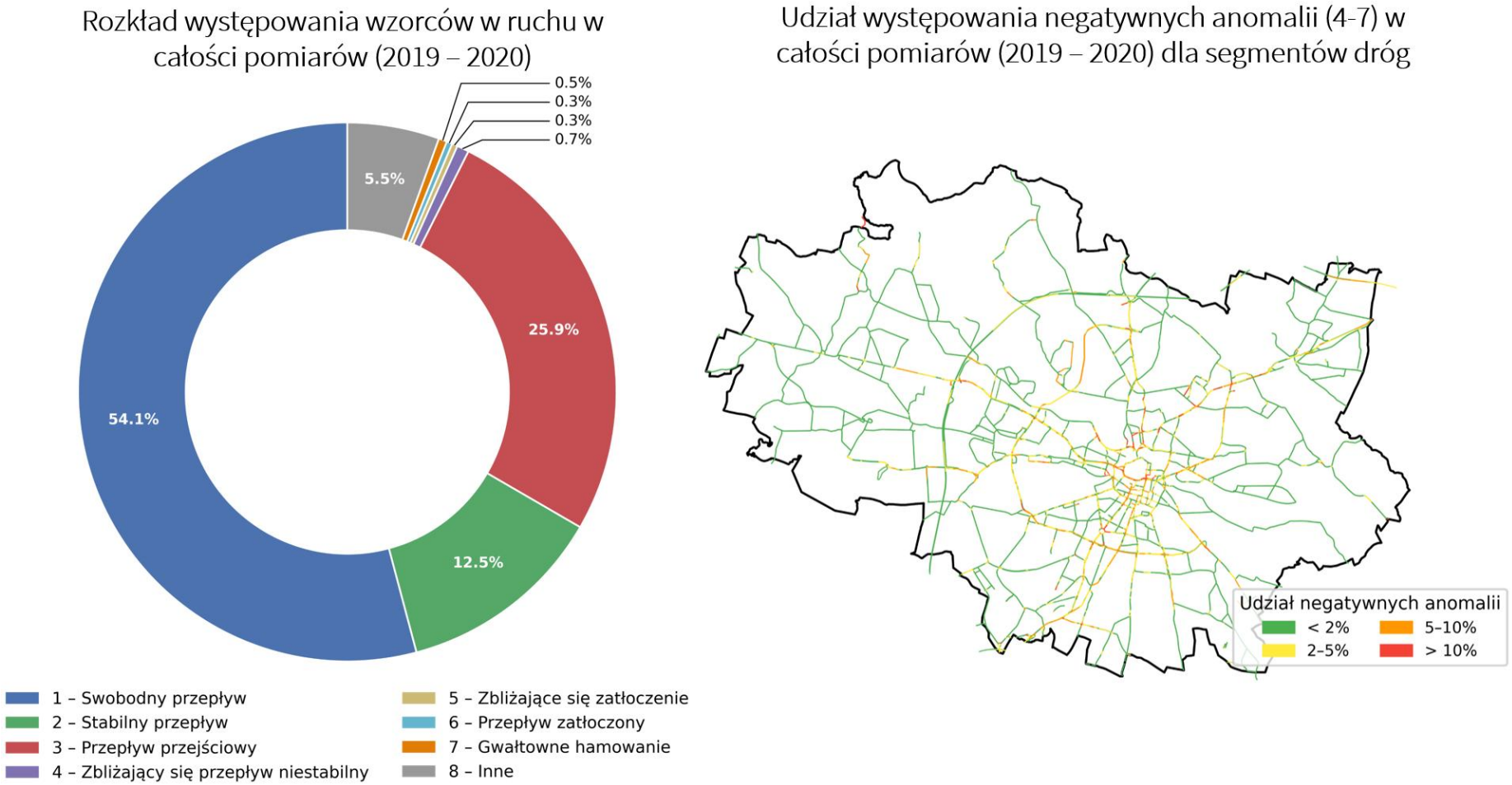
Rysunek 112. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Torunia (opracowanie własne)

Warszawa



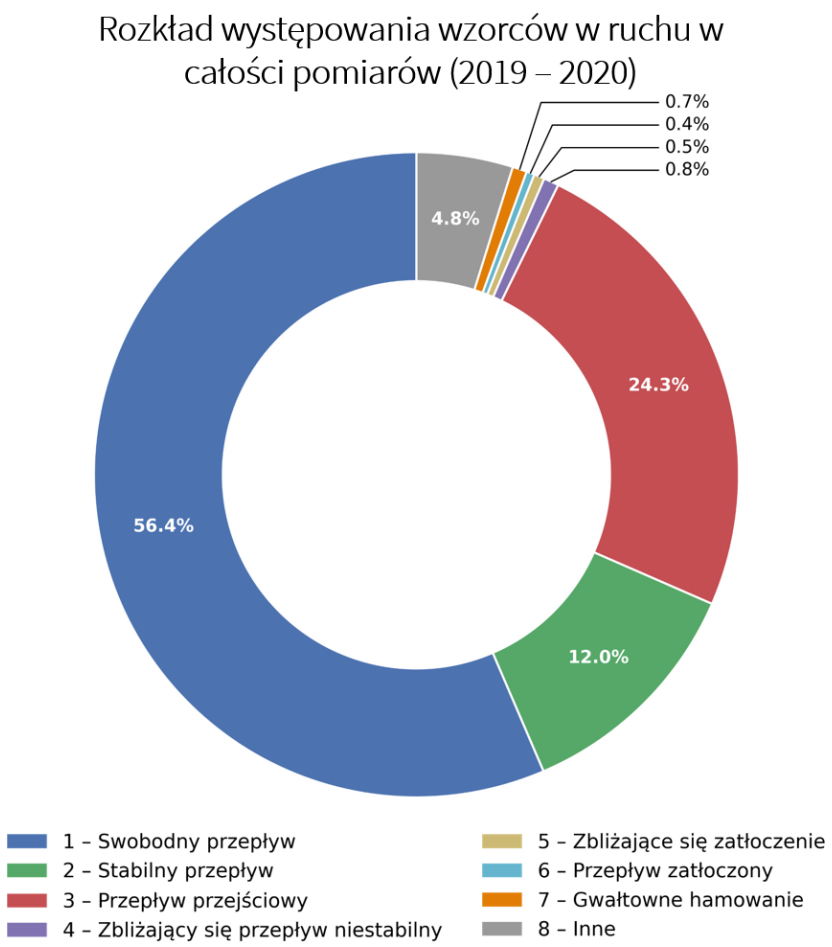
Rysunek 113. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Warszawy (opracowanie własne)

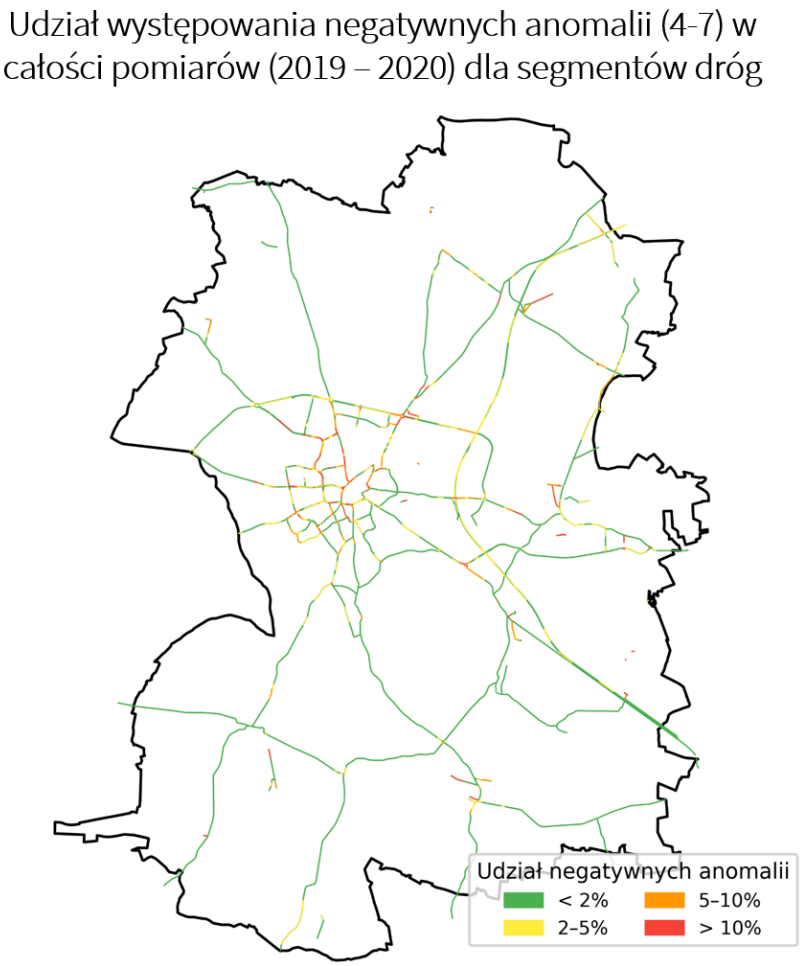
Wrocław



Rysunek 114. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Wrocławia (opracowanie własne)

Zielona Góra





Rysunek 115. Rozkład występowania wzorców w ruchu (lewa strona) oraz wizualizacja kartograficzna występowania negatywnych anomalii w ruchu dla całości pomiarów opracowanych w ramach scenariusza B4 dla Zielonej Góry (opracowanie własne)

Załącznik 4 – błędy środowiska Spark

Poniżej przedstawiono najczęściej występujące błędy środowiska SBD napotkanie w trakcie realizacji scenariuszy badawczych. Ilustrują one rodzaje problemów występujących w trakcie pracy z dużymi zbiorami danych przestrzennych w środowisku SBD CENAGIS: błędy konfiguracyjne, sieciowe, wysycenia zasobów obliczeniowych (pamięci RAM lub dyskowych) oraz awarie poszczególnych elementów klastra, w szczególności systemu plików HDFS. Prawidłowa diagnoza każdego ze wskazanych poniżej błędów wymagała analizy zarówno na poziomie aplikacji Spark, jak i samej infrastruktury. Rozwiązanie niektórych z opisanych poniżej błędów ograniczało się do optymalizacji przetwarzania, ale część z nich wymagała także działań na poziomie administratora systemu.

Błąd połączenia z HDFS

Błąd połącznia do HDFS objawiał się komunikatem o następującej treści:

java.net.ConnectException: Call From ... to srv-pw-r18-n32.cenagis.local:8020 failed on connection exception: Connection refused

...

Py4JJavaError: An error occurred while calling ... .Parquet.

Pojawia się on podczas zapisu lub odczytu zbiorów z przestrzeni HDFS. Najczęstszą przyczyną jego występowania była awaria HDFS powodująca jego czasową niedostępność w skutek czego nie był możliwy zapis ani odczyt danych. Jeżeli błąd wystąpił w trakcie przetwarzania dużego zbioru danych to w większości wypadków konieczne było jego ponowne przetworzenie. Rozwiązaniem tego problemu był restart usługi HDFS wykonany przez administratora systemu.

Brak bloku danych w HDFS

Brak bloku danych cechował następujący komunikat:

org.apache.hadoop.hdfs.BlockMissingException: Could not obtain block: ... file=/tmp/....Parquet

...

org.apache.spark.SparkException: Task failed while writing rows.

Powodem tego błędu była utrata spójności danych na HDFS spowodowana awarią systemu bądź przerwaną operacją zapisu danych. Taka sytuacja mogła wystąpić w przypadku, gdy zbiór wykorzystywany do analizy nie został zapisany w całości (np. przez problemy związane z siecią) lub jego zapis został przerwany. W takim wypadku należało usunąć dany zbiór i wykonać jego zapis ponownie

Tryb bezpieczny HDFS

Przykładowy komunikat podczas występowania tego błędu:

org.apache.hadoop.hdfs.server.namenode.SafeModeException: Cannot create directory /tmp/... . Name node is in safe mode.

The reported blocks ... needs additional ... blocks to reach the threshold ...

Jest to kolejny błąd związany z działaniem HDFS – w tym przypadku system wszedł w tryb bezpieczny nie pozwalający na zapis plików, co powoduje niemożność lub przerwanie wykonania wszelkich operacji kończących się zapisem plików wynikowych na HDFS. Tryb bezpieczny pozwala jedynie na odczyt danych. Rozwiązanie tego problemu wymagało interwencji administratora systemu w celu analizy spójności klastra HDFS i przywrócenia jego pełnej operacyjności

Brak miejsca na dysku w przestrzeni tymczasowej

Błąd ten występował w trakcie wykonywania wymagających operacji (złączanie bądź złączenie przestrzenne dużych zbiorów danych), gdzie wewnętrzne mechanizmy Spark przenosiły dane pośrednie na dysk lokalny maszyn *worker* ze względu na wysycenie całej ich pamięci operacyjnej. Wystąpienie takiego błędu objawiało się następującym komunikatem:

org.apache.spark.memory.SparkOutOfMemoryError: error while calling spill() on UnsafeExternalSorter : No space left on device

...

Job aborted due to stage failure...

W tym przypadku konieczna była optymalizacja samego algorytmu wykonania analizy w taki sposób, aby ograniczyć zapis danych pośrednich na dysku. Operacja ta nie tylko powoduje opisany powyżej błąd, ale może też znacznie spowalniać wykonanie operacji. Istnieje możliwość zwiększenia pamięci dostępnej dla Spark, ale nadal ograniczeniem jest wielkość zasobów dyskowych maszyn. Dodatkowym problemem był fakt, że w przypadku połączenia tego błędu z innymi awariami powodującymi nagłe przerwanie działania obliczeń Spark możliwa była sytuacja, gdzie dane tymczasowe nie zostały automatycznie usunięte co powodowało konieczność interwencji administratora systemu w celu dalszego prowadzenia obliczeń.

Zrywanie połączeń sieciowych

Ostrzeżenia o następującej konstrukcji:

WARN TransportChannelHandler: Exception in connection from /44.128.0.12:35716

java.io.IOException: Connection reset by peer

...

pojawiały się w przypadku długo trwających zadań obliczeniowych, a ich źródła wiązały się albo z chwilowymi przeciążeniami i innymi problemami sieciowymi albo z przekroczeniem limitów czasowych i zasobów po stronie maszyn *worker* w klastrze Spark. Najczęstszym rozwiązaniem tego problemu była optymalizacja algorytmów. W szczególności błąd ten występował często w związku ze zjawiskiem tzw. skrzywienia danych (ang. *data skew*), które powoduje, że nieproporcjonalnie duża część danych oraz obliczeń została przypisana tylko do jednego zadania w całym procesie obliczeniowym (np. partycja przestrzenna zawierała zbyt dużo danych). W przypadku gdy źródłem tego problemu były chwilowe awarie sieciowe najczęstszym rozwiązaniem był restart klastra Spark.

Przekroczenie limitu czasu dla operacji broadcast

Problem ten występował w przypadku, gdy w planie wykonania Spark pojawiła się operacja złączenia metodą *broadcast* (pełna kopia zbioru danych na wszystkie maszyny *worker*). Cechował go następujący komunikat:

Caused by: org.apache.spark.SparkException: Could not execute broadcast in 300 secs. You can increase the timeout for broadcasts via spark.sql.broadcastTimeout or disable broadcast join by setting spark.sql.autoBroadcastJoinThreshold to -1

W przypadku Platformy IT CENAGIS nie istniała jednak możliwość wyłączenia ani zwiększenia ograniczenia czasowego. Tak długi czas wykonywania operacji sugerował jednak niską wydajność nawet w przypadku, gdyby możliwość zwiększenia czasu wykonania istniała. W związku z tym konieczne było wymuszenie wykonania innego rodzaju złączenia poprzez manipulację samym zapytaniem w taki sposób, aby nie było ono wykonywane w trybie *broadcast*.