

**AGH**

**AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI AUTOMATYKI INFORMATYKI I  
INŻYNIERII BIOMEDYCZNEJ**

**KIERUNEK AUTOMATYKA I ROBOTYKA II STOPIEŃ  
SPECJALIZACJA INFORMATYKA W STEROWANIU I ZARZĄDZANIU**

**MODELOWANIE I ANALIZA PROCESÓW BIZNESOWYCH**

***Analiza relacji między zdarzeniami w kontekście algebry  
interwałowej Allena i logiki rozmytej.***

---

<i>L.p.</i>	<i>Członkowie grupy:</i>	<i>Nr. albumu</i>	<i>Adres email</i>
1	Bartosz Sroka	400490	sroka@student.agh.edu.pl
2	Artur Mzyk	400658	arturmzyk@student.agh.edu.pl
3	Dominik Czyżyk	401858	czyzyk@student.agh.edu.pl


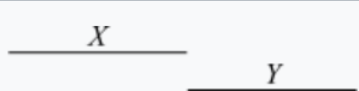
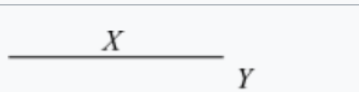
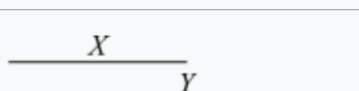
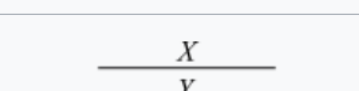
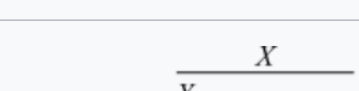
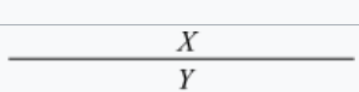
*Opiekun: dr inż. Krzysztof Kluza*

## **Spis treści**

Wstęp .....	2
Cel .....	3
Przygotowanie danych .....	3
Analizowane zbiory danych .....	4
Generator logów .....	4
Wizualizacja danych .....	5
Metody analizy logów .....	6
Metoda Trapezów .....	6
Metoda wielomianowa .....	8
Metoda macierzy relacji .....	10
Porównanie metod .....	11
Dodatkowe spostrzeżenia i analizy podczas realizacji projektu .....	12
Wnioski .....	13

## Wstęp

Algebra przedziałów Allena (ang. *Allen's interval Algebra*) stanowi ramy dla rozumowania o informacji czasowej w sposób jakościowy. W szczególności, wykorzystuje interwały, tj. pary punktów końcowych na osi czasu do reprezentowania jednostek odpowiadających działaniom, zdarzeniom lub zadaniom. Może również określać binarne relacje, takie jak poprzedzanie i nakładanie się, do kodowania możliwych konfiguracji pomiędzy tymi jednostkami. Algebra przedziałów Allena identyfikuje 13 podstawowych relacji.

Związek	Rysunek	Interpretacja
$X < Y$ $Y > X$		X ma miejsce przed Y ( <i>przed i po</i> )
$X m Y$ $Y m i X$		Y (x spełnia spełnia) ( / oznacza <i>I nverse</i> )
$X o Y$ $Y o i X$		X nakłada się Y ( <i>nakłada się</i> )
$X s Y$ $Y s i X$		X zaczyna Y ( <i>zaczyna</i> )
$X d Y$ $Y d i X$		X ma miejsce podczas Y (w <i>trakcie</i> )
$X f Y$ $Y f i X$		X kończy Y ( <i>kończy</i> )
$X = Y$		X jest równe Y

Rysunek 1. Relacje algebry Allena, źródło: [https://pl.frwiki.wiki/wiki/Alg%C3%A8bre\\_des\\_intervalles\\_d%27Allen](https://pl.frwiki.wiki/wiki/Alg%C3%A8bre_des_intervalles_d%27Allen)

Relacje przedstawione w na Rys. 1. są odrębne, wyczerpujące i jakościowe. Odrębne, ponieważ żadna para określonych przedziałów nie może być powiązana przez więcej niż jedną relację. Wyczerpujące, ponieważ każda para określonych przedziałów czasowych jest opisana przez jedną z relacji. Jakościowe, ponieważ pod uwagę brane są liczbowe przedziały czasowe.

13 relacji podstawowych opisuje relacje pomiędzy przedziałami określonymi. Interwały nieokreślone, których dokładna relacja może być niepewna, są opisywane przez zbiór wszystkich relacji podstawowych, które mogą mieć zastosowanie. Wówczas taka liczba może wzrosnąć nawet do  $2^{13}$  (8192) relacji.

Rachunek Allena znalazł zastosowanie w wielu akademickich i przemysłowych aplikacjach, które dotyczą najczęściej planowania i harmonogramów. Wykorzystuje się ją np. w systemach, które optymalizują czas pracy linii produkcyjnych pod kątem minimalizacji potencjalnych przestoju wynikających z czasu wykonywania się poszczególnych zleceń produkcyjnych. Ponadto algebra Allena znajduje zastosowanie w czasowych bazach danych oraz opiece zdrowotnej.

## Cel

Celem naszego projektu będzie zaproponowanie i zaimplementowanie różnych metod do znajdowania podstawowych relacji pomiędzy zdarzeniami w oparciu o algebrę Allena. Algorytmy zostaną ze sobą porównane pod kątem zakresu wykrywanych relacji, poprawności rezultatów dla różnych logów oraz czasu potrzebnego na ich wykonanie. Ponadto, stworzony generator logów będzie pozwalał na szybkie tworzenie sztucznych logów wyspecjalizowanych pod rozwiązanie konkretnego problemu. Całość zostanie zaimplementowana przy pomocy języka wysokiego poziomu jakim jest Python.

Wykryte relacje mogą być wykorzystane w rozpoznawaniu elementów procesu BPMN ale również w analizie zrównoleglania zdarzeń. Przeprowadzając taką analizę dla gotowego systemu, możemy zweryfikować, które ze zrównoleglonych zadań rzeczywiście wymaga tej operacji.

## Przygotowanie danych

Podstawowym krokiem, aby algorytmy przedstawione przez nas działały, jest odpowiednie przygotowanie danych wejściowych. Załadowane logi z plików .csv lub .xes muszą zawierać kolumny, które umożliwią nam określenie czasu rozpoczęcia i zakończenia zadań dla różnych ścieżek. Jeśli w naszych danych wejściowych mamy kolumnę wskazującą status, na przykład "start" i "complete", oraz mamy kolumnę *timestamp*, możemy przetworzyć dane za pomocą metody *dp.prepare\_data()* do pożądanej przez nas struktury. Przykładowe załadowane dane znajdują się na Rys. 2.

	org:resource	time:timestamp	concept:name	lifecycle:transition	case:concept:name
0	Call Centre Agent	2005-12-31 23:00:00+00:00	check if sufficient information is available	start	0
1	Call Centre Agent	2006-01-13 23:00:00+00:00	check if sufficient information is available	complete	0
2	Call Centre Agent	2006-01-13 23:00:00+00:00	register claim	start	0
3	Call Centre Agent	2008-01-29 23:00:00+00:00	register claim	complete	0
4	Claims handler	2008-01-29 23:00:00+00:00	determine likelihood of claim	start	0

Rysunek 2. Przykładowe załadowane dane

W wielu przypadkach dane wymagają indywidualnego podejścia i muszą zostać dodatkowo przanalizowane jednak końcowym wynikiem powinno być to, co jest widoczne na Rys. 3. Ważne jest aby załadowane dane w kolumnach określających czas zdarzeń były w formacie *pd.datetime()*. W przeciwnym wypadku metoda może zwrócić błąd.

	org:resource	concept:name	case:concept:name	start_timestamp	complete_timestamp
0	Call Centre Agent	check if sufficient information is available	0	2005-12-31 23:00:00+00:00	2006-01-13 23:00:00+00:00
1	Call Centre Agent	check if sufficient information is available	1	2005-12-31 23:00:00+00:00	2006-03-07 23:00:00+00:00
2	Call Centre Agent	check if sufficient information is available	10	2006-03-30 23:00:00+00:00	2006-04-09 23:00:00+00:00
3	Call Centre Agent	check if sufficient information is available	100	2008-07-26 23:00:00+00:00	2008-09-16 23:00:00+00:00
4	Call Centre Agent	check if sufficient information is available	101	2007-11-13 23:00:00+00:00	2008-01-19 23:00:00+00:00

Rysunek 3. Przygotowane dane

W podanym przypadku *org:resource* jest kolumną zawierającą instancje, *concept:name* jest to nazwa zadania, *case:concept:name* to numer ścieżki a pozostałe dwie kolumny określają utworzone chwile czasowe odpowiadające za rozpoczęcie i zakończenie zadania. Ładowany zbiór logów może posiadać więcej kolumn, jednak najważniejsze w kontekście analizy zdarzeń są przedstawione w przykładzie kolumny.

Istnieją również logi, które posiadają niewystarczającą ilość informacji o czasach wystąpienia zdarzeń. Większość logów, które przeanalizowaliśmy, zawierało jedynie informacje o zakończeniu zdarzeń, jednak nie było możliwe znalezienie czasu rozpoczęcia zadań. W takiej sytuacji możliwe są próby generacji czasu rozpoczęcia zadań, jednak bez dokładniejszej analizy logów nie będzie to dawało zbyt dobrych wyników. W naszej pracy skupiliśmy się głównie na logach, które są kompletne i możliwe do interpretacji bez ingerencji w strukturę dostępnych danych.

## Analizowane zbiory danych

Dobranie odpowiednich zbiorów danych było nie lada wyzwaniem. Większość dzienników zdarzeń dostępnych powszechnie w Internecie nie zawiera czasu zakończenia zadania (ang. *complete timestamp*), który był tutaj niezbędny. Udało się obejść to na dwa sposoby:

- opiekun dostarczył kilka dzienników zdarzeń, która zawierały czas zakończenia zadania,
- napisano funkcję do generacji sztucznych dzienników zdarzeń – jednak ma ona znaczne ograniczenia, tworzy tylko dzienniki zdarzeń dla dwóch zadań, które łączy ścisła relacja algebry Allena.

Dodatkowo, takie dzienniki zdarzeń trzeba było przeformatować, gdyż zawierały wiele zbędnych – z punktu widzenia problemu i wykorzystywanych metod – informacji. Schemat przygotowania danych został opisany w rozdziale „Przygotowanie danych”.

Niektóre z dostarczonych przez opiekuna dzienników zdarzeń była, kolokwialnie mówiąc, nieciekawa – występowały tam bardzo proste relacje, typu „meets” i „before”. Najbardziej zaawansowanym – a przez to najlepszym do analizy wykorzystywanych metod – okazał się dziennik zdarzeń znajdujący się w pliku *reviewing.csv*.

W pliku *reviewing.csv* znajdowały się logi, zawierające większość możliwych relacji Allena, ponadto w łatwy sposób można było przetworzyć dane i wykrywać w nich relację. Dostępne logi posiadają 5 wariantów dla, których można było sprawdzać relację między zdarzeniami.

Warto jeszcze wspomnieć, że niektóre dzienniki zdarzeń były przechowywane w plikach .xes, a inne w plikach .csv. Jednak po wstępnym załadowaniu dzienniki były zawsze przekształcane do tabeli *pd.DataFrame*, co pozwoliło na unifikowanie podejścia do obu typów plików.

## Generator logów

Generator logów według logiki Allena generuje event logi zawierające dwa zdarzenia: 'A' oraz 'B'. Generator tworzy *n* ścieżek wykonania, a każda ścieżka zawiera dwa zdarzenia 'A' oraz 'B', w których zdarzenia są wykonane w przedziale czasowym określonym przez daty minimalną i maksymalną z konfiguracji. Czas trwania pierwszego zdarzenia jest określony przez parametr '*first\_task\_duration*', a czas trwania drugiego zdarzenia przez parametr '*second\_task\_duration*'.

Dla każdej ścieżki czasy rozpoczęcia i zakończenia zdarzeń są losowane z rozkładu jednorodnego na określonym przedziale czasowym. 'A' jest losowany z rozkładu jednorodnego na przedziale czasowym między minimalną a maksymalną datą. Dla każdego rodzaju relacji wybrane są odpowiednie metodyki tworzenia relacji między zdarzeniami. Każda para tworzona jest z pewnym poziomem dywersyfikacji, który jest określony przez parametr '*level\_of\_differentiation\_of\_events*'. Im większy parametr (max 1) to zdarzenia występujące w ścieżkach są od siebie coraz bardziej różne.

Generowane są wyłącznie 6 z 13 możliwych relacji, ponieważ pozostałe są wyłącznie analogią wynikającą z odwrócenia wykonania zadań.

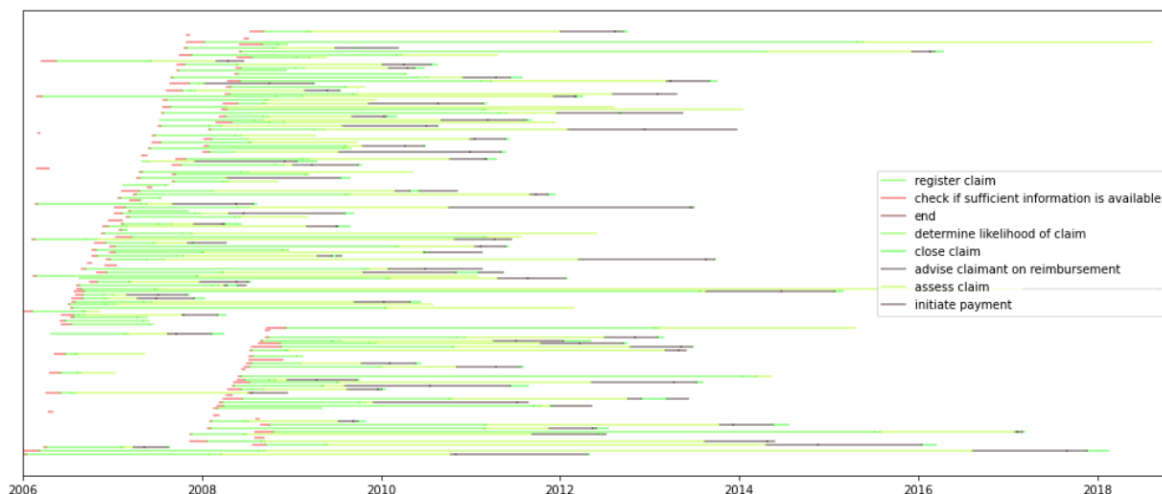
- *'meets'* – zdarzenia w relacji dotyczą się bez nakładania się na siebie.
- *'starts'* – pierwsze zdarzenie jest zawarta w drugim. Początki obu zdarzeń są równe. Analogiczną relacją byłoby *'ends'*, które nie zostało zaimplementowane, ale polegałoby na równym czasie zakończenia obu zdarzeń.
- *'before'* – relacja analogiczna do *'meets'*, natomiast zdarzenia występują w pewnym odstępie od siebie, bez nakładania.
- *'overlaps'* – relacja między dwoma zdarzeniami, w których oba zdarzenia się nakładają.
- *'contains'* – pierwsze zdarzenie zawiera się w całości w drugim.
- *'equal'* – relacja między dwoma zdarzeniami, które są identyczne.

Generacja logów zawierających wyłącznie jedną relację pozwala na analizę poprawności metod. Sprawdzamy czy dla prostych logów, rozpoznawane są prawidłowe relacje, wiedząc jakie zdarzenia wygenerowaliśmy.

## Wizualizacja danych

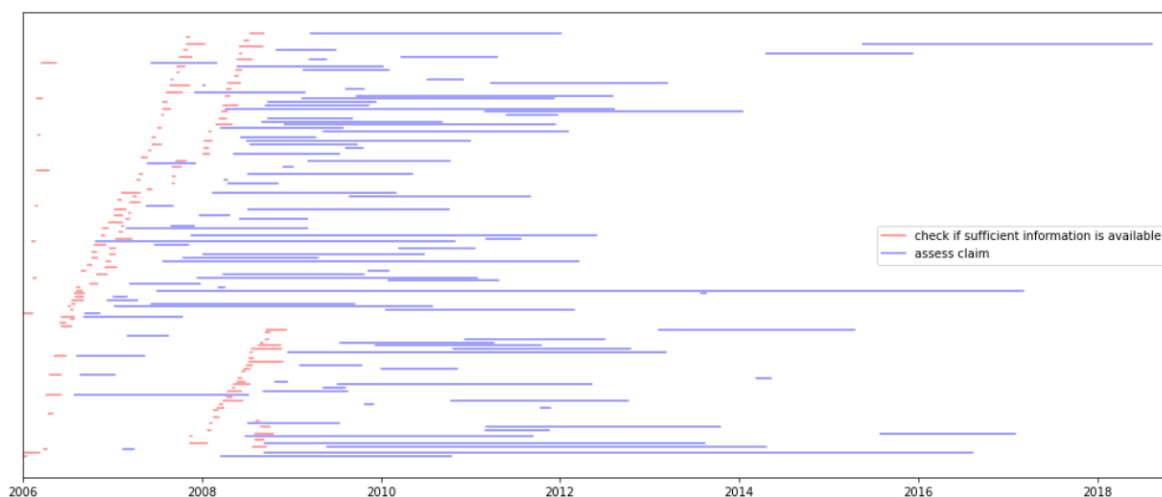
Dla małych ilości logów w łatwy sposób można przeprowadzić wstępną analizę zdarzeń w sposób wizualny, wyświetlając dostępne relacje, możemy obserwować czy w danych występuje nachodzenie się zdarzeń czy są one zupełnie rozłączne. Wizualizacja zdarzeń może być przydatna w weryfikacji poprawności implementowanych metod.

Na Rys. 4. został przedstawiony przykładowy przepływ zadań dla zbioru z pliku *exercise6.csv*.



Rysunek 4. Przepływ zadań dla zbioru z pliku *exercise6.csv*

Na wykresie są oznaczone kolorami odpowiednie zadania wraz z ich czasem trwania. Dla analizowanego zbioru mamy 821 ścieżek, które są wyświetlone jeden pod drugim na wykresie. Czas, w którym wykonały się wszystkie zadania, jest z przedziału od 2006 roku do 2019 roku. Na podstawie tego wykresu możemy wybrać interesujące nas zdarzenia, jednak bardziej przydatny ten typ wykresu jest w analizie zależności między dwoma konkretnymi zadaniami. Takie podejście zostało przedstawione na Rys. 5.



Rysunek 5. Przykładowy przepływ dla wybranych dwóch zadań

Dla wybranych zadań wykres staje się o wiele bardziej czytelny i widać, że w większości wypadków zadanie „*check if sufficient information is available*” wykonuje się przed „*assess claim*” przy czym zadanie drugie trwa kilka razy dłużej niż to pierwsze.

W celu wyświetlenia pierwszego wykresu, należy załadować dane do klasy *TrapezoidMethod* lub *PolynomialMethod* można wywołać wykorzystując metodę *.plot\_all\_tasks()*.

## Metody analizy logów

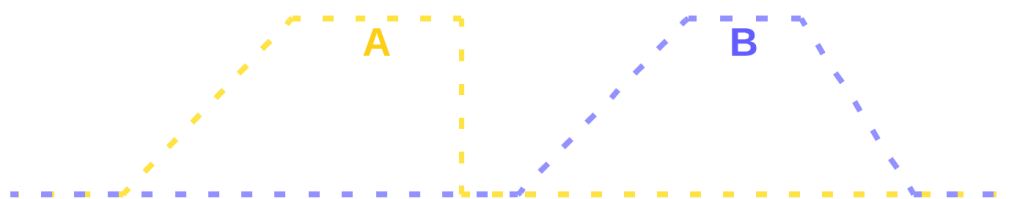
Wyżej przedstawione logi można interpretować w dwóch płaszczyznach: w poziomie i w pionie. Metoda w poziomie porównuje pary zdarzeń i określa ich relację dla każdej ścieżki, podczas gdy metody pionowe wykorzystują wszystkie informacje o czasach zdarzeń zbiorczo, a następnie wnioskuje o możliwej relacji. Metody pionowe są mniej dokładne z założenia, ale odznaczają się większą szybkością działania algorytmu dla dużych zbiorów danych.

W naszym rozwiązaniu przeanalizowaliśmy dwa rozwiązania przeglądu w pionie: *Metoda trapezów* oraz *Metoda wielomianowa* oraz jedno rozwiązanie w poziomie *Metoda macierzy relacji*.

### Metoda Trapezów

Metoda polega na określeniu relacji bazując wyłącznie na średnich i odchyleniu standardowym czasów rozpoczęcia i zakończenia zadań. Na podstawie obliczonych metryk tworzone są trapezy, w ten sposób, aby dół trapezu miał wartości  $(mean\_start\_timestamp - std\_start\_timestamp, 0)$  oraz  $(mean\_complete\_timestamp + std\_complete\_timestamp, 0)$ . Góra trapezu są to odpowiednio  $(mean\_start\_timestamp, ilość\_ścieżek)$  oraz  $(mean\_complete\_timestamp, ilość\_ścieżek)$ .

Dobranie takich parametrów pozwala na szerszy dół trapezu w przypadku krótszych, ale bardziej losowych zadań co bardziej odzwierciedla rzeczywiste relacje.



Rysunek 6. Przykładowa aproksymacja histogramów przepływu dwóch zadań metodą trapezów

Na Rys. 6. widać dwa zadania A i B. Zadanie A ma wartość *std\_complete\_timestamp* równą 0, ponieważ metoda w pierwszej kolejności powoduje wyrównanie zadań do końca pierwszego zadania. Na podstawie przedstawionego wykresu widać, że część wspólna pola pod wykresami zadań A i B jest równa 0. Oznacza to, że zadania są rozłączne, na podstawie wykresu można wywnioskować, że jest to relacja *'before'* jednak na podstawie obliczanych norm możemy osiągnąć wynik możliwych relacji z zakresu [*'before'*, *'meets'*].

Wynikiem analizy powierzchni pól są 2 miary. Jest to stosunek pola powierzchni części wspólnej do pola pierwszego zadania oraz do pola drugiego zadania. Na podstawie otrzymanych wartości można wnioskować o możliwościach wystąpienia zdarzeń. Dokładną analizę przedstawia poniższa funkcja, która wyklucza część przypadków dla analizowanych logów.

```
def return_possible_relation(p_value1: float, p_value2: float) -> List[str]:
    if p_value1 == 0 and p_value2 == 0:
        possible_relations = ['meets', 'before']
    elif p_value1 == 1:
        if p_value2 == 1:
            possible_relations = ['equals']
        else:
            possible_relations = ['contains', 'starts']
    elif (p_value1 == 1 and p_value2 <= 0.9) or (p_value1 <= 0.9 and p_value2 == 1):
        possible_relations = ['contains', 'starts']
    elif 0 < p_value1 <= 0.9 and 0 < p_value2 <= 0.9:
        possible_relations = ['meets', 'starts', 'overlaps']
    else:
        possible_relations = ['meets', 'starts', 'before', 'overlaps', 'contains', 'equals']
    return possible_relations
```

W najgorszym przypadku metoda zwróci pełny zestaw analizowanych relacji. Jednak, jeśli jakaś relacja się wyróżnia to jesteśmy w stanie ją wybrać.

Metoda wywoływana jest dla załadowanych danych. Należy zaimportować metodę:

```
from trapezoidal_method import TrapezoidMethod
```

Następnie załadować dane do zaimportowanej klasy określając nazwy wymaganych kolumn dla przygotowanego dataframe'u.

```
Texercise6 = TrapezoidMethod(logs=df,
                              case_id_column_name='case:concept:name',
                              task_column_name='concept:name',
                              instance_column_name='org:resource',
                              start_timestamp_column_name='start_timestamp',
                              complete_timestamp_column_name='complete_timestamp')
```

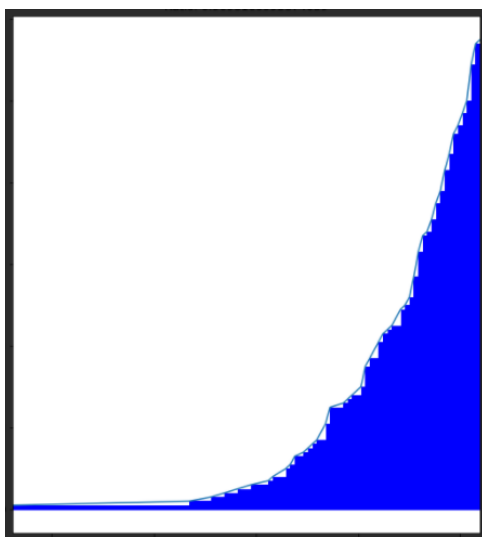
Dla załadowanych danych wywołujemy już gotową metodę z wybranymi kolumnami oraz parametrami wyświetlającymi odpowiednie elementy algorytmu:

```
Possibles_relation = Texercise6.core_metod(first_task=,
                                           second_task=,
                                           instance=,
                                           plot_results=True,
                                           plot_steps=True,
                                           print_results=True)
```

*first\_task* oraz *second\_task* są to wybrane z możliwych danych zadania, dla których badamy relację. *instance* jest to parametr, który może zostać podany w sytuacji, kiedy chcemy przeanalizować relację dla wybranej instancji. Parametr *plot\_results* wyświetla wykres trapezów końcowych elementów. *plot\_steps* wyświetla kroki równania zadań oraz ich wygląd. Natomiast *print\_results* jest to parametr, który wypisuje najważniejsze parametry zawarte w rozwiązywaniu problemu.

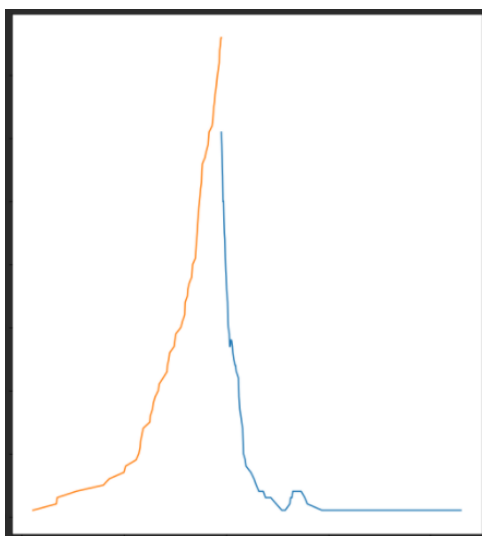
## Metoda wielomianowa

Metoda polega na określeniu relacji, bazując na histogramie opisującym częstość występowania danego zadania w danym przedziale czasowym – inaczej mówiąc, bazując na rzutowaniu przepływu zadań na oś czasu. Następnie taki histogram jest aproksymowany z wykorzystaniem funkcji sklejanych trzeciego stopnia (ang. *cubic splines*), co pozwala na szybsze obliczenie pola pod takim histogramem. Gdy znane są funkcje aproksymujące histogramy dla obu zadań, można wyznaczyć część wspólną obszarów pod krzywymi i obliczyć pożądane współczynniki.



Rysunek 7. Przykładowa aproksymacja funkcjami sklejanymi

Na Rys. 7. Przedstawiona została przykładowa aproksymacja histogramu przepływu zadania funkcjami sklejanymi.



Rysunek 8. Przykładowe zestawienie na jednym wykresie dwóch aproksymacji dla relacji "before"



Na Rys. 8. zostało przedstawione przykładowe zestawienie na jednym wykresie dwóch aproksymacji dla relacji „before”. Z wykresu „gołym okiem” widać, że właśnie taka relacja łączy dwa zadania. Jednakże pole części wspólnej obszarów pod krzywymi daje jedynie informacje o tym, że jest to jedna z dwóch relacji: „before” i „meets”.

Wynikiem analizy powierzchni pól są dwie miary:

- stosunek pola powierzchni części wspólnej obszarów pod krzywymi do pola powierzchni obszaru pod funkcją aproksymującą dla pierwszego zadania,
- stosunek pola powierzchni części wspólnej obszarów pod krzywymi do pola powierzchni obszaru pod funkcją aproksymującą dla drugiego zadania.

Na podstawie otrzymanych wartości można wnioskować o możliwościach wystąpienia zdarzeń – podejście takie samo jak w metodzie trapezów, gdyż po otrzymaniu powyższych współczynników podejście obu metod się unifikuje.

Wynikiem analizy powierzchni pól są 2 miary. Jest to stosunek pola powierzchni części wspólnej do pola pierwszego zadania oraz do pola drugiego zadania. Na podstawie otrzymanych wartości można wnioskować o możliwościach wystąpienia zdarzeń. Dokładną analizę przedstawia poniższa funkcja, która wyklucza część przypadków dla analizowanych logów.

Metoda wywoływana jest dla załadowanych danych. Należy zaimportować metodę:

```
from polynomial_method import PolynomialMethod
```

Następnie trzeba załadować dane do zaimportowanej klasy, określając nazwy wymaganych kolumn dla przygotowanej tabeli *pd.DataFrame*.

```
Polynomial_exercise6 = PolynomialMethod(df,
                                         case_id_column_name='case:concept:name',
                                         task_column_name='concept:name',
                                         instance_column_name='org:resource',
                                         start_timestamp_column_name='start_timestamp',
                                         complete_timestamp_column_name='complete_timestamp')
```

Dla załadowanych danych wywołuje się już gotową metodę z wybranymi kolumnami oraz parametrami wyświetlającymi odpowiednie elementy algorytmu:

```
possible_relations = polynomial_exercise6.core_method(task1=,
                                                       task2=,
                                                       instance=,
                                                       plot_results=True,
                                                       plot_steps=True,
                                                       print_results=True)
```

Parametry *task1* oraz *task2* są to nazwy wybranych z możliwych zadań, dla których bada się relację. Parametr *instance* może zostać podany w sytuacji, kiedy chce się przeanalizować relacje dla wybranej instancji. Parametr *plot\_results* pozwala wyświetlić wykres funkcji aproksymujących. Parametr *plot\_steps* pozwala wyświetlić kroki równania zadań oraz ich wygląd. Natomiast parametr *print\_results* pozwala wypisać najważniejsze parametry wykorzystane podczas rozwiązywania problemu.

## Metoda macierzy relacji

Metoda polega na znalezieniu dominującej relacji pomiędzy zdarzeniami. Dla każdego trace'a i sąsiadujących ze sobą zdarzeń algorytm określa występującą pomiędzy nimi relację i zapisuje ją w odpowiednie miejsce macierzy relacji (ang. *relation matrix*). Sama macierz relacji to pythonowy dataframe, którego wierszami i kolumnami są nazwy tasków. Na przecięciu przykładowych dwóch takich zdarzeń znajduje się lista znalezionych pomiędzy nimi relacji. Przykładowa macierz relacji została przedstawiona na Rys. 9.

	check if sufficient information is available	register claim	determine likelihood of claim	assess claim	initiate payment	advise claimant on reimbursement
check if sufficient information is available	-	[mi, mi, mi, mi, mi, mi, mi, m, ...]	[>, >, >, >, >, >, >, >, >, >, ...]	[>, >, >, >, >, >, >, >, >, >, ...]	-	-
register claim	[m, m, m, m, m, m, m, m, ...]	-	[mi, mi, mi, mi, mi, mi, mi, m, ...]	[>, >, >, >, >, >, >, >, >, >, ...]	[>, >, >, >, >, >, >, >, >, >, ...]	[>, >, >, >, >, >, >, >, >, >, ...]
determine likelihood of claim	[<, <, <, <, <, <, <, <, ...]	[m, m, m, m, m, m, m, m, ...]	-	[mi, mi, mi, mi, mi, mi, mi, m, ...]	[>, >, >, >, >, >, >, >, >, >, ...]	[>, >, >, >, >, >, >, >, >, >, ...]
assess claim	[<, <, <, <, <, <, <, <, ...]	[<, <, <, <, <, <, <, <, ...]	[m, m, m, m, m, m, m, m, ...]	-	[mi, >, mi, >, >, mi, >, mi, >, ...]	[>, mi, >, mi, >, >, mi, >, mi, >, ...]
initiate payment	-	[<, <, <, <, <, <, <, <, ...]	[<, <, <, <, <, <, <, <, ...]	[m, <, m, <, <, m, <, m, <, ...]	-	[mi, m, mi, m, m, >, m, mi, mi, >, m, >, m, ...]
advise claimant on reimbursement	-	[<, <, <, <, <, <, <, <, ...]	[<, <, <, <, <, <, <, <, ...]	[<, m, <, m, m, <, m, m, <, ...]	[m, mi, m, mi, mi, <, mi, mi, m, m, <, mi, <, ...]	-

Rysunek 9. Przykładowa macierz relacji

Użytkownik może manipulować działaniem metody poprzez dwa parametry. Są nimi: liczba rozważanych zdarzeń sąsiadujących (*numer\_of\_following\_tasks\_considered*) oraz próg uznania relacji za dominującą (*relation\_threshold*). Pierwszy parametr określa pomiędzy iloma następującymi po sobie zdarzeniami algorytm ma szukać relacji. Pozwala to na jej wykrycie w sytuacji, gdy wcześniejszy, lewy task jest skorelowany z więcej niż jednym następującym po nim zdarzeniami np. start bramki równoległej. Drugi parametr, jak sama nazwa sugeruje, pozwala manipulować wartością uznania relacji za zasadną. Przykładowy *relation\_threshold=0.7* oznacza, że dana relacja musi występować przynajmniej w 70% przypadków pomiędzy tymi dwoma zdarzeniami.

Przebieg działania metody składa się z kilku kroków. Najpierw inicjalizowana jest pusta macierz relacji. Kolejno zdarzenia są grupowane po *case\_id*, czyli po trace'ie, do którego należą. Następnie iterujemy po wszystkich pogrupowanych taskach i znajdujemy pomiędzy nimi relacje (funkcja *find\_allens\_algebra\_relation\_between\_tasks*).

```
def find_allens_algebra_relation_between_tasks(left_task, right_task):
    if left_task["complete_ts"] < right_task["start_ts"]:
        return "<;>"
    elif left_task["complete_ts"] == right_task["start_ts"]:
        return "m;mi"
    else:
        if left_task["start_ts"] == right_task["start_ts"]:
            if left_task["complete_ts"] == right_task["complete_ts"]:
                return ";;="
            else:
                return "si;si"
        else:
            if left_task["complete_ts"] < right_task["complete_ts"]:
                return "o;oi"
            elif left_task["complete_ts"] == right_task["complete_ts"]:
                return "fi;f"
            else:
                return "di;d"
```

Na koniec, jeśli dla danej pary zdarzeń pewna relacja dominuje, wówczas uznawana jest ona za zasadną i zapisywana do ostatecznego dataframe'a zwracanego przez metodę.

Metodę można wywołać w prosty sposób importując klasę będącą implementacją metody

```
from relation_matrix_method import AllenRelationsFinderNaiveMethod
```

oraz wywołując odpowiednie funkcje klasowe.

```
allen_relations_finder_naive_method =  
AllenRelationsFinderNaiveMethod(logs=el_df_cleaned)  
  
relation_matrix =  
allen_relations_finder_naive_method.create_relation_matrix(number_of_following_tasks_considered=3)  
  
founded_relations =  
allen_relations_finder_naive_method.find_relations(relation_threshold=0.7)
```

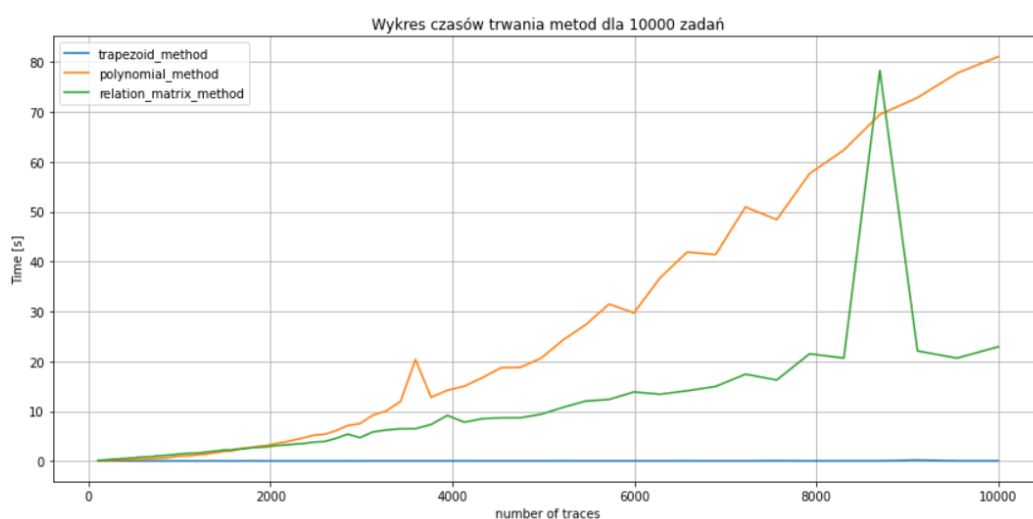
## Porównanie metod

Jak zostało wspomniane w poprzednich rozdziałach, metoda trapezów i metoda wielomianowa działają podobnie - skupiają się na aproksymacji histogramów przepływów zadań i dążą do uzyskania odpowiednich współczynników opisujących stosunek pól pod krzywymi. Niestety nie są one metodami dokładnymi i pozwalają jedynie na ograniczenie potencjalnych relacji łączących dwa zadania, aniżeli na dokładne wywnioskowanie, jaka relacja łączy owe zadania. Natomiast metoda macierzy relacji jest dokładna, gdyż przechodzi w pętli po wszystkich instancjach i wyznacza dla nich relację, a następnie na podstawie zbiorczych danych wyznacza jedną relację.

Zdecydowanie najszybsza jest metoda trapezów, co sprawia, że wiezie prym nad metodą wielomianową, gdyż obie dają jedynie informacje o tym, które relacje na pewno nie występują. Można się spierać, czy lepiej zastosować wolniejszą, ale dokładną metodę macierzy relacji. Najlepszym rozwiązaniem jest zastosowanie kaskady obydwu metod, czyli:

1. Ograniczenie listy potencjalnych relacji metodą trapezów.
2. Wyznaczenie dokładnej relacji metodą macierzy relacji.

Na Rys. 10. przedstawiono wykresy czasowe dla wszystkich trzech metod.



Rysunek 10 Wykres czasów trwania algorytmu dla analizowanych metod

## **Dodatkowe spostrzeżenia i analizy podczas realizacji projektu**

Poniżej znajdują się analizowane przypadki oraz możliwości rozwoju algorytmów. Niektóre z nich są przydatnymi spostrzeżeniami, które pomogły rozwinąć oprogramowanie. Niektóre są elementami, które w trakcie analizy zostały odrzucone lub pominięte podczas realizacji zadania jednak mogą się okazać przydatnym punktem wyjścia w rozwoju oprogramowania.

### **Analiza równania zdarzeń**

Początkowy projekt zakładał wyrównanie zadań do początku pierwszego zdarzenia. Takie rozwiązanie pozwala na analizę logów w pionie, ponieważ wszystkie zdarzenia znajdują się w podobnym zakresie czasu. Późniejsza analiza przedstawiona na poniższych wykresach pokazuje, że najlepszym rozwiązaniem, które przynosi lepsze wyniki, jest wyrównanie zadań do końca pierwszego zadania lub początku kolejnego zadania. Takie podejście pozwala nam uniknąć nakładania się trapezów w sytuacjach, gdy takiej relacji nie ma.

### **Wykorzystanie funkcji sklejanych**

Początkowo w metodzie wielomianowej próbowano aproksymować histogram przepływu zadania pojedynczym wielomianem, jednak okazało się to dość niepraktyczne, gdyż taki histogram ma bardzo nietypowy kształt – ciężko było dobrać odpowiedni stopień wielomianu i aproksymacja nie była zadowalająca, co potwierdzało porównanie pola pod krzywą i pola pod histogramem liczonego przy pomocy pól prostokątów.

Wykorzystanie funkcji sklejanych pozwoliło na dokładniejszą aproksymację i zwolniło z obowiązku doboru stopnia wielomianu.

### **Losowe wybieranie zdarzeń do analizy z logów**

Losowe wybieranie zdarzeń powoduje, że w zależności od liczności danej relacji w zbiorze możemy uzyskać inne wyniki. W przypadku występowania relacji dominującej w zbiorze danych wybranie losowych wartości może polepszyć wykrytą relację lub lekko pogorszyć wyniki. Jednak dla małych wartości nie mają one większego wpływu na wynik algorytmów.

### **Zmiana podejścia metody – eliminacja niemożliwych zdarzeń**

Początkowo metody aproksymacji funkcją oraz metoda trapezów miały zwracać wynikowe relacje. Jednak ze względu na brak możliwości rozróżnienia zdarzeń w rzeczywistych przypadkach, metody zostały przerobione, aby zwracały możliwe relacje, czyli eliminują wszystkie niemożliwe relacje. Takie podejście pozwala zawęzić pole poszukiwań relacji w zbiorze. Ze względu na to, że przegląd poziomy i tak określa każdą relację to nie można wykorzystać możliwych relacji do przyspieszenia działania algorytmu.

### **Wykorzystanie metod eliminacji relacji w metodzie macierzy relacji**

Czas wykonania metody trapezów pozwala na odkrycie możliwych relacji dla 10000 logów w czasie około 0.025 sekundy, gdzie metodzie macierzy relacji zajmuje to około 30 sekund. Ze względu na charakterystykę metody, wykonuje się ona w szybkim czasie. Przyszłościowo można spróbować wykorzystać tę metodę do wstępnych analiz logów a kolejno stworzyć metodę, która będzie mogła z tego skorzystać.

## Wykorzystanie t-norm

Funkcja t-norma to funkcja zdefiniowana w następujący sposób:

$$T: [0,1] \times [0,1] \rightarrow [0,1],$$

która spełnia jeszcze kilka warunków.

W metodzie macierzy relacji wyznacza się częstość występowania danej relacji, co po znormalizowaniu daje wartości z przedziału  $[0,1]$  dla każdej relacji, co skłaniałoby do zastanowienia się, czy nie można tutaj wykorzystać w jakiś sposób t-norm. Jednakże ostatecznie nie znaleziono żadnego zastosowania.

## Wnioski

Udało się pomyślnie doprowadzić do końca procedurę wyznaczania relacji trzema odrębnymi metodami, które jako dane wejściowe przyjmują dziennik zdarzeń w postaci pliku .csv lub .xes i nazwy dwóch zadań, a ich danymi wyjściowymi jest informacja o potencjalnych relacjach bądź dokładna relacja wiążąca dwa zadania. Każda z nich dała jakieś wymierne rezultaty.

Można raczej z czystym sumieniem założyć, że metoda trapezów jest zdecydowanie lepsza niż metoda wielomianowa – jest znacznie szybsza, a daje podobne wyniki. Natomiast można się spierać w porównaniu jej z metodą macierzy relacji. Jednakże, tak jak już wcześniej wspomniano, najlepiej połączyć obydwie metody.

Największą trudnością tego projektu było zdobycie odpowiednich dzienników zdarzeń, które zawierałyby czas zakończenia zadania. Implementacja metod była dosyć przyjemna – każdy z członków zespołu miał swoją własną.

Krokami dalszego rozwoju mogłoby być:

- poprawienie wydajności metod – zwłaszcza wielomianowej, jeśli to możliwe,
- wyprowadzanie dodatkowych parametrów metodami w pionie tak, aby możliwe było wyznaczenie dokładnej relacji – takich jak odstępy na osi czasu pomiędzy aproksymacjami (pozwoliłoby to na odróżnienie relacji „meets” i „before”),
- próba wnioskowania o procesie na podstawie otrzymanych relacji algebry Allena - zbadanie wpływu wykrywanych relacji na postać schematu blokowego rozważanego procesu (pozwoliłoby to na jego odtworzenie na podstawie logów składających się z wielokrotnie zapuszczanych instancji).