

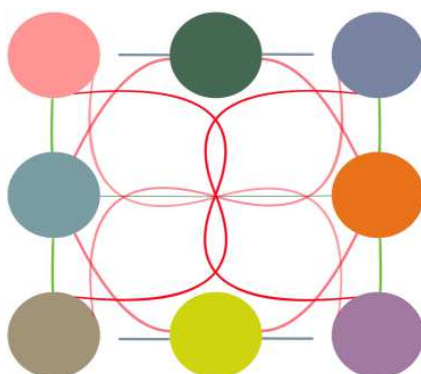


POLITECHNIKA POZNAŃSKA

**Wydział Informatyki
Instytut Informatyki**

inż. Łukasz Myśliński

System wizualizacji błędów odwzorowania w metodzie MDS



Praca magisterska napisana pod kierunkiem:
dr inż. Robert Susmaga



Politechnika Poznańska
WYDZIAŁ INFORMATYKI

Poznań, dnia 29.06.2013 r.

OŚWIADCZENIE

Ja, niżej podpisany Łukasz Myśliński, student Wydziału Informatyki Politechniki Poznańskiej oświadczam, że przedkładaną pracę dyplomową (magisterską) pt. „System wizualizacji błędów odwzorowania w metodzie MDS” napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób.

Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało się nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

.....

podpis



Spis treści

1.	Wprowadzenie	5
1.1	Wstęp	5
1.2	Przegląd istniejących rozwiązań	8
1.2.1	GGvis	9
1.2.2	PerMap	11
2.	Skalowanie wielowymiarowe	13
2.1	Podstawy teoretyczne MDS	14
2.2	Metody pozyskiwania danych i przykłady zastosowania metody MDS	15
2.3	Model danych	18
2.4	Opis klasycznej metody MDS	18
2.4.1	Przykład obliczeń	19
2.5	Obliczanie błędów skalowania	21
3.	Projekt systemu wizualizacji danych wielowymiarowych	24
3.1	Analiza zapotrzebowań	25
3.2	Wymagania pozafunkcjonalne	27
3.3	Opis technologii, narzędzi i bibliotek	28
3.3.1	Technologia WPF	28
3.3.2	Język C#	28
3.3.3	Narzędzie Syncfusion Metro Studio	29
3.3.4	Narzędzie Inkscape	29
3.3.5	Narzędzie IcoFX	29
3.3.6	Narzędzie Wolfram Alpha	29
3.3.7	Narzędzie Microsoft Visual C# 2010 Professional	29
3.3.8	Biblioteka MahApps.Metro	30
3.3.9	Biblioteka DotNumerics	30
3.3.10	Biblioteka NLog	30
3.4	Opis struktury systemu	30
3.5	Opis implementacji	32
3.5.1	Generowanie linii przerywanych	38
3.5.2	Generowanie linii falistych o zadanej długości	39
3.6	Interfejs systemu	42
3.6.1	Ekran główny	42
3.6.2	Ekran ładowania danych	42
3.6.3	Ekran dekompozycji	44
3.6.4	Ekran wynikowy	45
3.6.5	Ekran wizualizacji	45
3.7	Testowanie systemu	46
3.7.1	Testy użytkowników końcowych	47
4.	Badane zbiory	48
4.1	Odtwarzanie topologii na podstawie odległości	48
4.2	Odtwarzanie podobieństwa między obiektami	51
5.	Podsumowanie	54
5.1	Weryfikacja użytkowa	54
5.2	Rozbudowa systemu	56
6.	Bibliografia	57
7.	Dodatki	61
7.1	Spis ilustracji	61
7.2	Spis tabel	61
7.3	Diagramy klas	62
7.4	Modele przypadków użycia	66
7.4.1	Proces obsługi danych	66
7.4.2	Proces wizualizacji	68
7.5	Algorytmy	70
7.6	Dodatkowe ilustracje	71



Spis oznaczeń

- MDS – skalowanie wielowymiarowe (ang. Multi-Dimensional Scaling, MDS)
- SVD – rozkład według wartości szczególnych (ang. Singular Value Decomposition)
- PCA – analiza głównych składowych (ang. Principal Component Analysis, PCA)
- EVD – dekompozycja za pomocą wartości własnej macierzy (ang. Eigen-Value Decomposition, EVD)
- CSV – format przechowywania danych w plikach tekstowych (ang. Comma Separated Values - wartości rozdzielone przecinkiem)
- Plugin - wtyczka (ang. plug-in, add-on), czyli dodatkowy moduł rozszerzający możliwości macierzystej aplikacji czy też większego programu
- Perceptual Mapping – technika graficzna używana przez branżę marketingową do wizualizacji percepcji i opinii klientów na temat produktów
- Zettabajt – jednostka określająca tryliard bajtów danych (skrót ZB)
- Eksabajt – jednostka określająca trylion bajtów danych (skrót EB)
- IDC – Międzynarodowa Korporacja Danych (ang. International Data Corporation, IDC)
- Png – format pliku graficznego wysokiej jakości
- Xps – format pliku dokumentu, zawierającego obiekty w postaci wektorów

Streszczenia

Praca opisuje przebieg budowy systemu wizualizacji danych wykorzystujący metodę MDS. W wyniku skalowania wielowymiarowego powstaje błąd, który system prezentuje szczegółowo poprzez odległości nieeuklidesowe między obiektami.

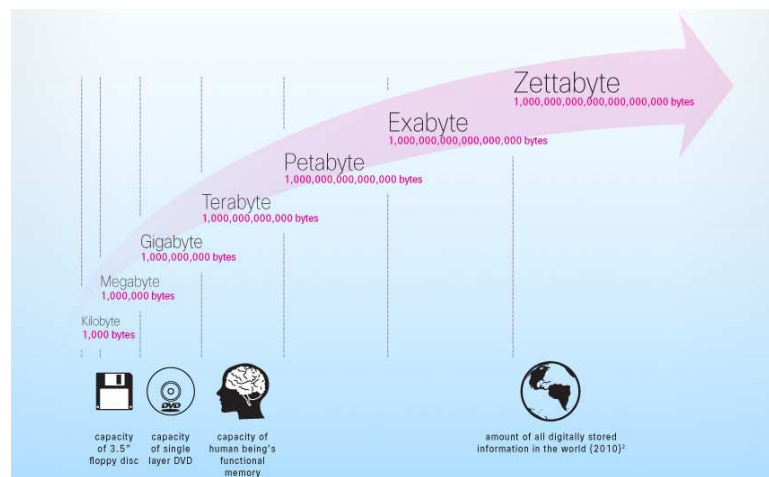
This work describes the process of building a visualisation system of data based on MDS method. As a result of multidimensional scaling, there is an error which the system is representing by non-euclidean distances between objects.



1. Wprowadzenie

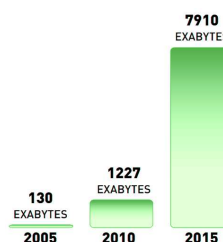
1.1 Wstęp

Rzeczywistość technologiczna komputerowa postępuje z roku na rok w praktycznie każdej dziedzinie życia. Zwiększają się możliwości sprzętowe cyfrowych urządzeń, co idzie w parze z ciągłymi obniżkami cen. Powoduje to coraz powszechniejszy dostęp do mocnych obliczeniowo narzędzi zarezerwowanych dotychczas dla nielicznych. Co więcej, ilość ludzi wymieniających między sobą informacje ciągle rośnie. To wszystko sprawia, iż dane, które są zapisywane na różnych nośnikach danych od baz danych do taniejących dysków twardych, zalewają nas z każdej strony. Codziennie tworzenie danych bije swój własny rekord. W 2010 roku całkowita ilość danych wygenerowana przez świat przekroczyła po raz pierwszy ponad jeden zettabajt [RIYA 2013].



Rysunek 1.1 Część infografiki ukazującej skalę jednostki zettabajt [WEBS 2010]

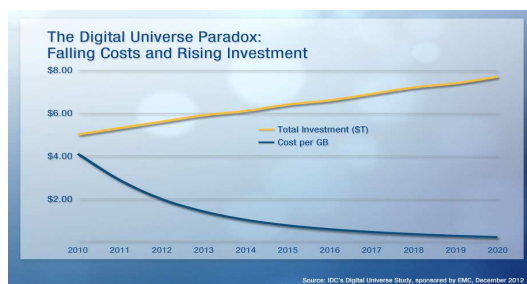
Od ponad pięciu lat IDC publikuje raporty o nazwie „The Digital Universe” sponsorowane przez korporację EMC zajmującą się szeroko pojętym przechowywaniem danych. Wynika z nich, że do końca roku 2015 świat wygeneruje około osiem zettabajtów danych. [IDC 2011]



Rysunek 1.2 Porównanie wygenerowanych ilości danych w raporcie „The Digital Universe” [IDC 2011]

Rodzi to kolejne wyzwania, przed którymi stoją naukowcy od statystyków po inżynierów oprogramowania, którzy wdrażają rezultaty kolejnych badań. Wkraczanie tak ogromnej ilości danych w nasze codzienne życie wyznacza nowe kierunki rozwoju tej gałęzi nauki. Koszty potrzebne na tego typu inwestycje w szeroko pojęte zarządzanie takimi danymi stale się podnoszą, ponieważ nie potrafimy sobie poradzić z analizą tylu danych.

Generuje to swoisty paradoks informacji, ponieważ wzrost informacji nie idzie w parze z naszymi możliwościami odczytywania tych danych.



Rysunek 1.3 Stosunek ceny wszystkich inwestycji (żółta linia) do analizy, utrzymywania i zarządzania jednym gigabajta danych (niebieska linia) [IDC 2012]

Potrzebna jest metoda, która będzie elastyczna, szybka, efektywna i precyzyjna. W tym momencie warto sobie zadać pytanie czy istnieje technika, która sprostałaby tym trudnym wymaganiom? Z pewnością jedną z ważniejszych jest wizualizacja danych, która dzięki ludzkiej percepcji uzyskuje dodatkową cechę jaką jest kreatywność w procesie analizy danych. Dodatkowo, dzięki bezpośrednim interakcjom, mamy możliwość większej przenikliwości i szybszego wyciągnięcia wniosków. Niniejsza praca ma za zadanie połączyć te cechy przy wykorzystaniu najnowszej technologii.

Zmysł wzroku dostarcza człowiekowi najwięcej informacji. Dwadzieścia bilionów neuronów jest przeznaczonych do analizy obrazów, dostarczając mechanizmów odnajdywania wzorców [WARE 2006]. Skoro metoda została wybrana, należałoby zadać pytanie: ile wymiarów zaprezentować tak, aby człowiek mógł w pełni i komfortowy sposób odczytywać wizualizację danych?

Z pomocą znów przychodzi biologia. Otóż, siatkówka oka jest wycinkiem sfery do której światło dociera z zewnątrz poprzez soczewkę. Tym samym, obraz powstający na niej i przekazywany do mózgu jest dwuwymiarowy. Z kolei, człowiek widzi świat poprzez parę oczu, czyli docierają do nas dwa obrazy dwuwymiarowe. Aż osiemdziesiąt procent wszystkich informacji odbieramy

poprzez ten narząd, a tylko dziesięć procent kory mózgowej zaangażowanej jest w interpretację tych informacji. Dlatego to co postrzegamy jako trzy wymiary jest tak naprawdę rekonstrukcją przez nasz mózg nałożonych na siebie dwóch obrazów dwuwymiarowych. W takim razie, czy istnieje jakaś przeszkoda w wizualizacji więcej niż trzech wymiarów? Niestety, o ile obiekty trzy wymiarowe jesteśmy w stanie sobie wyobrazić poprzez ich obserwację z różnych stron, to przestrzeni o wyższym niż trzeci wymiar nie jesteśmy w stanie nawet odnieść do rzeczywistości [JAMR 2001].

Dobra technika wizualizacji danych powinna prezentować dane w taki sposób, aby człowiek mógł z łatwością wyciągnąć wnioski czy uprościć proces decyzyjny. Omawiane podejście jest szczególnie wartościowe wówczas, gdy nie mamy wystarczającej wiedzy na temat badanego problemu i gdy cele eksploracji nie są jasno określone [KEIM 2002]. Jeżeli dodamy do tego kolejny fakt dotyczący dużych danych, czyli są to w większości dane wielowymiarowe, które są trudne do interpretacji. W praktyce okazuje się, że takie dane posiadają wiele atrybutów, które są mocno ze sobą powiązane. Co więcej, aby uzyskać pełen obraz relacji pomiędzy danymi czy po prostu jakieś zachowanie wynikające z nich, wystarczy niewielki podzbiór tych danych. Wychwycenie takich wyników dla wszystkich danych jest po prostu niemożliwe albo bardzo trudne do uzyskania. Stąd coraz większe znaczenie ma technologia służąca do redukcji wymiarowości.

Zmniejszanie ilości wymiarów polega na przekształcaniu danych wielowymiarowych (czyli o dużej ilości atrybutów) do przestrzeni o sensownie mniejszym wymiarze. Z powodu ograniczeń, o których wyżej wspomniano najczęściej wynosi on dwa. [GRAM 2009].

Z kolei, skalowanie wielowymiarowe pozwala na redukcję atrybutów w wyniku której powstaje utrata pewnej ilości informacji. W idealnym przypadku wspomniana strata wynosi prawie zero. W tym obszarze bardzo dobrze spisuje się MDS. Jest to zbiór metod służących do wizualizacji danych wielowymiarowych. Obecnie żadna aplikacja realizująca tą metodę nie ukazuje szczegółowo wspomnianej wyżej niedokładności między poszczególnymi obiektami. Biorąc pod uwagę liczne zastosowania metody MDS jest to kolejny krok do podejmowania trafniejszych decyzji i precyzyjniejszej wizualizacji.



Celem tej pracy jest zaprojektowanie i zaimplementowanie systemu wizualizacji błędów odwzorowania danych wielowymiarowych w metodzie MDS. Uwzględniającego niedokładności w generowanych położeniach obiektów z wykorzystaniem odległości nieeuklidesowych.

Struktura pracy jest następująca. W rozdziale 2 przedstawiono szczegółowy opis metody MDS na podstawie obecnej literatury. Rozdział 3 jest poświęcony budowie systemu. Z kolei rozdział 4 zawiera opis badanych zbiorów oraz oceny jakości wizualizacji zaproponowanego systemu. Rozdział 5 stanowi podsumowanie pracy, w której są przedstawione uzyskane efekty oraz etapy rozwoju systemu.

1.2 Przegląd istniejących rozwiązań

Rozdział ten został poświęcony porównaniom aplikacji wykorzystujących metodę MDS. Oprogramowanie zostało podzielone na trzy kategorie: interaktywne, z grafiką o wysokiej rozdzielczości oraz bez grafiki o wysokiej rozdzielczości.

Tabela 1.1 Spis oprogramowania oferującego skalowanie wielowymiarowe [MMDS 2007]

Nazwa	Typ	Licencja
GGVis	Interaktywna	Darmowa
PerMap	Interaktywna	Darmowa
Alscal	Grafika wysokiej jakości	Komercyjne
NewMDSX©	Grafika wysokiej jakości	Komercyjne
Proxscal	Grafika wysokiej jakości	Komercyjne
SAS	Grafika wysokiej jakości	Komercyjne
Statistica	Grafika wysokiej jakości	Komercyjne
Systat	Grafika wysokiej jakości	Komercyjne
Guttman-Lingoes	Bez grafiki wysokiej jakości	Darmowa
KYST	Bez grafiki wysokiej jakości	Darmowa
Multiscale	Bez grafiki wysokiej jakości	Darmowa

Z uwagi na dostępność programów oraz ich popularność zostaną opisane tylko dwa poniższe programy:

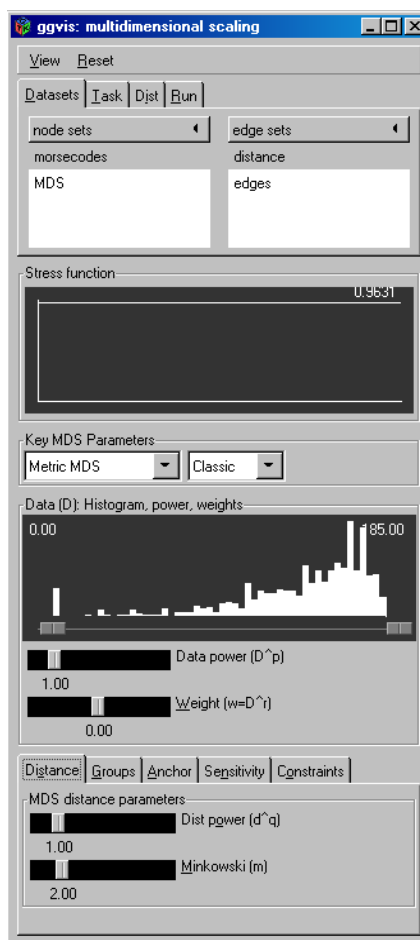
- GGvis – plugin, który jest częścią darmowej aplikacji GGobi służącej do wizualizacji oraz eksploracji danych wielowymiarowych,
- PerMap – darmowa aplikacja (skrót od PERceptual MAPping Software), która oferuje metryczne oraz niemetryczne techniki MDS redukujące wielokrotne relacje między preferencjami do postaci obrazów tworząc percepcyjne mapy (po ang. „perceptual maps”).

Więcej opisów aplikacji oraz dokładniejsze ich porównanie można znaleźć w dodatku publikacji [BORG 2005].

1.2.1 GGvis

Moduł jest uruchamiany z sekcji narzędzia po załadowaniu danych w aplikacji GGobi. W głównym oknie mamy do wyboru następujące metody MDS:

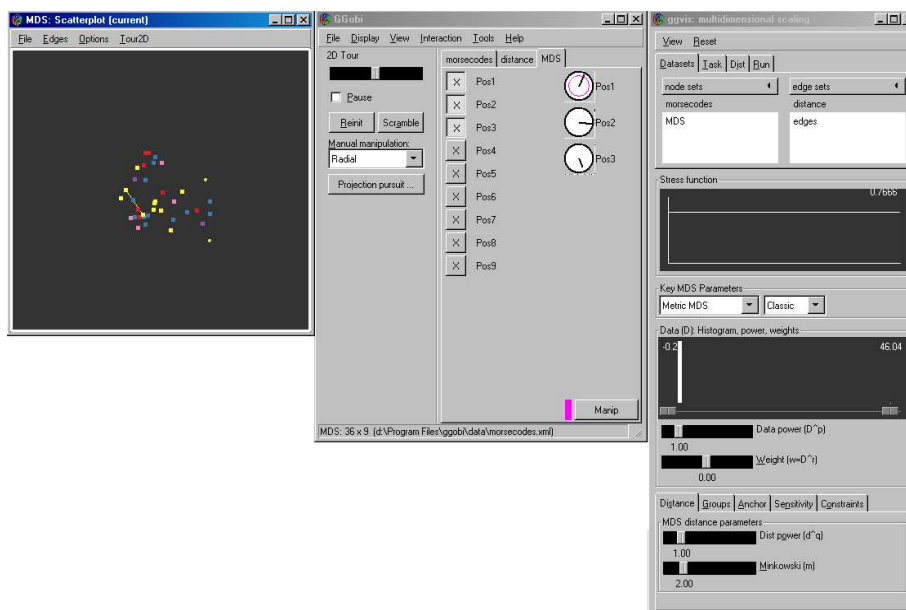
- metryczny oraz niemetryczny w wersji Torgerson-Gower (tzw. klasyczny)
- metryczny oraz niemetryczny w wersji Kruskal-Shepard



Rysunek 1.4 Główny widok dodatku ggvis

Główną zaletą tej aplikacji jest sposób implementacji klasycznego skalowania poprzez iteracyjną minimalizację funkcji kosztu, co sprzyja użyciu wag oraz brakujących wartości w danych w przeciwieństwie do algorytmu EVD. Dodatkowo można sterować etapami znalezionych progów po zakończeniu znalezienia odpowiedniej konfiguracji współrzędnych. Dzięki temu możemy zobaczyć animację w czasie rzeczywistym, w jaki sposób obiekty znajdują swoje idealne położenie. Kolejną cechą interaktywnej wizualizacji jest przeciąganie dowolnego obiektu lub jej grupy za pomocą myszki, ukazując animację dopasowywania sąsiednich obiektów. Co więcej, dla danych w przestrzeni trzywymiarowej można użyć rotacji w celu obserwacji dopasowania obiektów pod każdym kątem. Z kolei zaś dla wymiarów większych niż trzy, system umożliwia uruchomienie tzw. „grand tour”, czyli animacji w postaci wycieczki obiektów. Dodatek posiada też wiele udogodnień w celu polepszenia wyniku optymalizacji. Można to wykonać za pomocą łączenia linią najbardziej zbliżonych obiektów. Taki stopień interaktywności jest nazywany analizą wrażliwości (ang. sensitivity analysis), ponieważ skupia się na tym, w jaki sposób zmienić parametry, aby rozwiązanie było najbardziej optymalne.

Przechodząc do wad programu, należy wziąć pod uwagę mnogość opcji porozrzucanych po oknach aplikacji, przez co obsługa nie jest zbyt intuicyjna i prosta. Co więcej aplikacja potrafi często zamknąć się z nieznanego powodu na systemie Microsoft Windows XP SP3®.



Rysunek 1.5 Widok w module ggvis po uruchomieniu metody MDS



Aplikacja jest dostępna za darmo na systemy X Window System TM, UNIX®, Linux® i została przeniesiona także na system Microsoft Windows™. Więcej informacji dostępnych jest w dokumencie [BUJA 2001], który stanowił podstawę dla tego podrozdziału oraz na stronie [GGOBI].

1.2.2 PerMap

Fundamentalnym celem programu PerMap jest odkrywanie ukrytej struktury, która jest zawarta często w złożonych danych. Przyjmuje na wejście programu nieujemne dane, które:

- posiadają wagi, czyli każdy obiekt może być bardziej lub mniej preferowany,
- są symetryczne, czyli wartość z wiersza i , kolumny j odpowiada wartości z wiersza j i kolumny i ,
- niekompletne, czyli nie posiadające żadnych wartości,
- są trójkątne, czyli wartości są uzupełnione tylko do przekątnej danych,
- są prostokątne, czyli wartości są uzupełnione w każdym wierszu i kolumnie,
- mogą być obiektami (do tysiąca na raz), a każdy obiekt może mieć do stu atrybutów.

Obsługa tak różnorodnych danych z pewnością jest dużą zaletą tej aplikacji. Dodatkowo obsługuje różne metryki między innymi „city block” wyliczany ze wzoru:

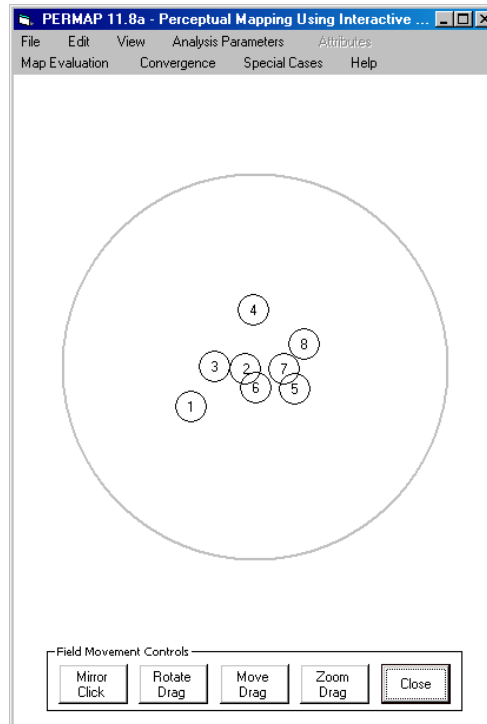
$$\sum_{j=1}^k |a_j - b_j|$$

gdzie a i b to punkty, a k określa ilość wymiarów. Różni się tym od Euklidesowego dystansu (odzwierciedlającego możliwie najkrótszą drogę między punktami), że jest sumą poszczególnych odległości między wymiarami. Jednakże oba modele bazują na metryce Minkowskiego:

$$\left(\sum_{j=1}^k |a_j - b_j|^p \right)^{\frac{1}{p}}$$

gdzie $p \geq 1$. Skąd dla $p = 1$ otrzymujemy wzór na odległość „city block”, z kolei $p = 2$ dla dystansu Euklidesowego.

Aplikacja posiada też różne miary obliczania błędów, do wyboru w trybie rzeczywistym. Rozwiązuje problemy do ośmiu wymiarów i umożliwia nakładanie warunków brzegowych. Inaczej mówiąc, są to zespoły warunków, które muszą zostać spełnione, aby jednoznacznie wybrać rozwiązanie.



Rysunek 1.6 Zrzut ekranu z programu PerMap

Co więcej jak można z łatwością zauważyć na rysunku 1.6, zezwala także na interakcję wizualizacji w trybie rzeczywistym poprzez na przykład manipulację położenia poszczególnych obiektów (zbliżanie, oddalanie, rotację czy przesunięcie obiektów). Kolejną ciekawą rzeczą jest funkcja wyłączania i włączania wybranego obiektu z metody MDS poprzez umieszczenie go w specjalnym kontenerze o nazwie parking. Dodatkowo można dany obiekt zablokować na danej pozycji. Korzystając z tych funkcji można z pewnością zweryfikować stabilność uzyskanych rozwiązań.

Pomimo wielkiej popularności i około dwudziestoletniej historii program przestał być rozwijany i obecnie ciężko jest go uruchomić na systemie powyżej Microsoft Windows XP. Wersja 11.8a została dostosowana do uruchomienia pod Microsoft Windows 7. Tak jak w przypadku aplikacji GGVIS program potrafi zakończyć swoje działanie bez powodu. Więcej szczegółowych informacji można znaleźć w dokumencie [HEAD 2010], który stanowi cenne źródło dla tego podrozdziału.

2. Skalowanie wielowymiarowe

Skalowanie wielowymiarowe to metodologia, której zadaniem jest umieszczenie obiektów o wysokiej ilości wymiarów do przestrzeni o niższym wymiarze (we wstępie ustalono, że wynosi on dwa) zachowując przy tym określoną metrykę między nimi oraz jak najlepsze ich odwzorowanie. Człowiek określając otaczającą go rzeczywistość w sposób bardziej świadomy operuje wymiarami, czyli atrybutami, które to otoczenie opisuje [BIELA 1992]. Co więcej, potrafimy w sposób intuicyjny przeprowadzić skalowanie tych danych. Mózg codziennie zmierza się z tym problemem, wyodrębniając z wielowymiarowego sensora danych wejściowych, składających się z dziesiątek tysięcy nerwów narządów zmysłów [TENE 2000]. Często nie zdajemy sobie z tego sprawy, kiedy chcemy wywnioskować coś znaczącego dla wielu nieznających informacji. To jest właśnie cel skalowania.

Procedury skalowania, aby tego dokonać używają określonych zasad (modeli) do przydzielania wartości liczbowych dla różnych cech danych uważanych przez obserwatora za jakościowe. Ich wynikiem są pomiary nazywane wartościami skalowanymi. Z kolei pomiar jest procesem przydzielania wartości liczbowych (algorytm) dla atrybutów obserwowanych, które są uważane przez obserwatora za ilościowe. Proszę zauważyć, że skalowanie jak i pomiar produkują miary za pomocą pewnych wartości liczbowych określających obserwowane cechy w danych okolicznościach. Różnica polega na tym, że skalowanie pochodzi z pomiarów jakościowych, a pomiary z danych ilościowych. W takim razie skąd wiemy czy dana obserwacja jest jakościowa czy ilościowa? Nie wiemy, dlatego musimy założyć jedną lub drugą. W rzeczywistości różnica nie zawiera się w obserwacji, a w umyśle obserwatora [YOUN 1984].

Techniki skalowania można podzielić na dwie kategorie ze względu na dane, które są skalowane. Mogą to być dane podobieństw lub odległości pomiędzy obiektami. Dodatkowo można je podzielić ze względu na przestrzeń, w której się znajdują, czyli liniowa i nieliniowa (lokalnie liniowa).

Do nieliniowych algorytmów redukcji wymiarów należy Isomap (więcej w [TENE 2000]), maximum variance unfolding [WEIN 2004], locally linear embedding (więcej w [SAUL 2000]), laplacian eigenmaps (szczegóły dostępne są w publikacji [BELK 2002]). Są to nowe metody, które wynikły z rozwoju

takich technologii jak rozpoznawanie ludzkiej twarzy, ludzkiej mowy, ręcznego pisanie czy też naturalnego języka. Doskonale sprawdzają się wszędzie tam, gdzie potrzebne jest odwzorowanie nieliniowych kształtów. Oczywiście im więcej relacji między obiektami, które trzeba zachować tym bardziej forma odbiega od pożądanej. Kluczem w tych algorytmach jest zachowanie odpowiedniego balansu w identyfikacji, gdzie większość informacji spoczywa. [GASH 2009].

Z kolei, jeśli chodzi o liniowe metody to ich rozwój rozpoczął się już w latach osiemdziesiątych. Najbardziej dynamicznie rozwijaną był i jest właśnie MDS ze względu na swoją prostotę i dobrą dokumentację. Jego główną zaletą nad takimi popularnymi metodami jak SVD czy PCA jest jego wysoka dokładność w reprezentowaniu odległości między różnymi przypadkami, szczególnie jeśli są to wielowymiarowe obiekty. PCA posiada wadę jaką jest używanie tylko zmiennych ilościowych. Dlatego konieczne jest posiadanie pełnych danych z obserwacji, w przeciwnym wypadku nie da się użyć tej metody. Natomiast, metoda SVD może być używana w ograniczonych przypadkach. Metoda ta przyjmuje tylko dane pochodzące z obiektów, których współrzędne są nieujemne. MDS jest pozbawiony tych wszystkich ograniczeń, dlatego został wybrany do realizacji skalowania danych wielowymiarowych, o czym więcej w następnym rozdziale.

2.1 Podstawy teoretyczne MDS

Parafrazując słynne chińskie przysłowie „jeden obraz wart więcej niż tysiąc słów” na „jeden obraz wart więcej niż tysiąc liczb” można ogólnie streścić wszystkie rodzaje metody MDS. Pierwsza dostępna technika MDS powstała dzięki pracom Torgerson’a [TORG 1952],[TORG 1958] i Gower’a [GOWE 1966]. Opisują one skalowanie klasyczne i dostarczają pierwsze rozwiązanie metryczne. Dlatego też często jest nazywana metodą „skalowania Torgerson’a” lub „skalowanie Torgerson’a i Gower’a”. W ogólności przyjmuje ona za model danych wejściowych macierz podobieństwa lub odległości powstałą poprzez zastosowanie na obiektach danej metryki (miary odległości), które określają stopień zgodności między nimi. Wynikiem jest taka konfiguracja obiektów w przestrzeni o zadanej liczbie wymiarów, dla której odległości pomiędzy obiektami są możliwie najlepiej odwzorowane. Skalowaniu podlega więc oryginalny rozmiar danych z zachowaniem zarówno ich własności topologicznych

jak i metrycznych. Szczegółowy opis kroków algorytmu klasycznego skalowania znajduje się w rozdziale 2.4 Opis klasycznej metody MDS.

Przełom w badaniach nastąpił w latach sześćdziesiątych XX wieku za sprawą owocnej pracy [SHEP 1962]. Shepard rozważał w niej problem prezentowania obiektów takich jak kolory, dźwięki, figury, twarze, znaczenie wyrazów i inne, jako punkty w przestrzeni, w taki sposób, aby zachować między obiektami podobieństwo. Ostatecznie rozwiązał to Kruskal proponując procedurę minimalizującą w swojej pracy nad wersją niemetryczną metody MDS [KRUS 1964]. To właśnie od tego czasu zaczęły się pojawiać liczne aplikacje. Metoda ta opiera się na fakcie, że dane mogą być interpretowane tylko w sensie porządku lub rankingu. Skalowanie niemetryczne (porządkowe) różni się tym od metrycznego, że jest zachowywany tylko ranking między obiektami. Jest on często stosowany w psychologii i pokrewnych dziedzinach, gdzie większe znaczenie ma macierz bliskości składająca się z danych porządkowych.

Skalowanie wielowymiarowe może być stosowane do wielu typów danych. W gruncie rzeczy, każda macierz danych nadaje się do analizy o ile jej elementy oznaczają siłę lub stopień relacji między obiektami reprezentowanymi przez kolumny i wiersze. Takie dane oznaczane są przez „relacyjne” dane jak na przykład korelacje, dystanse, bliskości, podobieństw, liczne wskaźniki skale klasyfikacji czy też macierze preferencji. Z tego wynika, że metody skalowania zostały zaprojektowane z myślą o wielu typach danych. Włączając w to symetryczne i asymetryczne macierze, prostokątne lub kwadratowe macierze, macierze z brakującymi elementami, jednakowe i nierównomierne powtarzające się dane w macierzach i inne tego typu [YOUN 1987].

2.2 Metody pozyskiwania danych i przykłady zastosowania metody MDS

Procedury MDS zakładają, że na wejściu znajduje się już przygotowana macierz podobieństw lub odległości. Jednakże warto sobie uzmysłowić w jaki sposób one powstają oraz w jakich obszarach mogą być zastosowane. Istnieje wiele sposobów na zbieranie danych. Metody te można podzielić na dwie kategorie.

Pierwsza z nich dotyczy bezpośrednich ocen – większość analizowanych danych, z których korzysta MDS używa tego typu metody. Polega ona tym, że badane osoby przydzielają liczbowe wartości podobieństwom (lub



odległościom) dla każdej pary obiektów lub przypisują ranking w odniesieniu do ich podobieństw (lub odległości). Zawiera ona z kolei takie przykładowe podkategorie:

- ocena – najbardziej oczywisty sposób polega na tym, żeby badana osoba oceniła stopień podobieństwa (lub odległości) między obiektami w czasie i odpowiedniej skali. Najlepiej, aby skala posiadała tyle kategorii na ile jest to możliwe uwzględniając efektywność opinii. Na przykład obecnie dość popularnym zjawiskiem jest tworzenie strony na portalu społecznościowym przez różnego typu znane firmy. Przeważnie prowadzi je osoba z firmy marketingowej, która ma możliwość tworzenia różnego rodzaju ankiet i zbierania opinii. W ten sposób może zebrać oceny na temat produktu firmy na tle dziesięciu innych konkurencji. Wykona to umieszczając ankietę, w której każda opcja będzie porównaniem produktu danej marki, którą reprezentuje z marką konkurenta i wskaże skalę od jeden do dziesięć (jeden oznacza brak podobieństwa, a dziesięć – bardzo podobne),
- uszeregowanie – badana osoba ustala kolejność odległości między obiektami,
- sortowanie – grupa badanych osób otrzymuje zbiór obiektów do posortowania do dowolnie wielu grup, w kategorii niepodobieństwa (lub podobieństwa) między nimi w taki sposób, że obiekt z tej samej grupy jest bardziej podobny do pozostałych, aniżeli obiekty w osobnych zbiorach. W bardziej formalny sposób metoda jest stosowana na kartach z wypisanymi obiektami, które później są grupowane w odpowiednie stosy przez badane osoby. Metoda jest bardzo prosta, nawet dla wielu obiektów. Z tego powodu często jest używana w naukach odnoszących się do społeczeństwa. Została użyta do badania symboli fonetycznych w języku japońskim.

Druga z nich dotyczy metod pośrednich. Nie wymagają one przypisywania wartości liczbowych elementom z macierzy wejściowej, ponieważ pochodzi ona z zastosowania innych miar:



- dane subiektywne – jest rzeczą oczywistą, że im bardziej podobne są do siebie obiekty tym większe prawdopodobieństwo pomyłki. Obiekty są prezentowane w parach, żeby badana osoba mogła zdecydować czy są takie same czy różne. Miarą podobieństwa jest proporcja, która wynika z ocen tych samych obiektów w momencie pojawienia się oceny, z których wynika, że obiekty są różne. W ten sposób można otrzymać wyniki dla obiektów, które bardzo łatwo można pomylić np. kod Morse’a,
- częstotliwość współwystępowania – miarą podobieństwa w tym wypadku jest częstotliwość występowania pewnej cechy na przykład u różnych osób przy ich opisywaniu. W ten sam sposób działa to również, kiedy dwie osoby dzielą tą samą cechę. Im więcej mają takich współwystępujących cech tym bardziej obiekty są podobne do siebie niż do innych,
- opóźniona reakcja – oznacza przedział czasowy potrzebny do poróżnienia dwóch podobnych do siebie obiektów. Polega to na tym, że mierzony jest czas reakcji na szybkie stwierdzenie przez badaną osobę czy obiekty są podobne czy różne. W ten sposób można zbadać na przykład podobieństwo między wyrazami twarzy,
- interakcje społecznościowe – jak sama nazwa wskazuje miarą podobieństwa jest częstotliwość interakcji społecznościowych, która wskazuje na zażyłość między ludźmi. W obecnych czasach z pewnością możemy to odnieść do wielu portali społecznościowych, w których młodzi ludzie skupiają wielkie ilości znajomych,
- korelacje między profilami – idea tej metody polega na tym, że jeżeli obiekty są podobne na różnych atrybutach, które ich dotyczą, to w ogólności są one bardzo bliskie sobie. Przykładem może być znalezienie podobnych do siebie konsumentów po produktach, które zakupili w danym sklepie lub wyłonienie grupy bliskich osób na podstawie filmów, muzyki czy jedzenia, które lubią.

Podsumowując istnieje wiele różnych metod pozyskiwania danych dla macierzy wejściowej metody MDS. Badacz na podstawie rodzaju badań i kontekstu pytania może zdecydować, którą metodę wybierze. Bardziej szczegółowe dane na temat

metod pozyskiwania danych znajdują się w publikacji [TAKA 2009], która posłużyła za źródło tego rozdziału. Więcej przykładów z podziałem na dyscypliny naukowe znajduje się w publikacji [LEEU 2000].

2.3 Model danych

Model danych definiuje w jaki sposób dane w macierzy wejściowej są mapowane do odległości w n wymiarowej metodzie MDS. Z uwagi na popularność i analityczne podejście dalsza część pracy będzie odnosić się tylko do klasycznej wersji metody MDS.

Najbardziej popularnym i naturalnym jest euklidesowy model funkcji dystansu. Jest on łatwo parametryzowany w kartezjańskim układzie współrzędnych. Za x_{ir} przyjęto współrzędne punktu i w wymiarze r . Z kolei d_{ij} oznacza odległość euklidesową między punktami i oraz j , obliczoną na podstawie wzoru

$$d_{ij} = \left\{ \sum_{r=1}^R (x_{ir} - x_{jr})^2 \right\}^{\frac{1}{2}}$$

gdzie R określa liczbę wymiaru reprezentującej przestrzeni. W metodzie MDS zestaw wartości współrzędnych $\{x_{ir}\}$ dla $i=1, \dots, n$ (gdzie n oznacza liczbę punktów) i $r=1, \dots, R$ są określone w taki sposób, że zbiór d_{ij} obliczony z x_{ir} jest tak podobny jak to możliwe do zaobserwowanej danej bliskości między obiektami.

Dystans d w przestrzeni metrycznej X musi zachowywać jej aksjomaty:

- dla każdego $i, j \in X, d_{ij} > 0$ i $d_{ij} = 0$ tylko w przypadku $i = j$,
- dla każdego $i, j \in X, d_{ij} = d_{ji}$,
- dla każdego $i, j, k \in X, d(x, k) \leq d(x, y) + d(y, k)$ - nierówność trójkąta.

2.4 Opis klasycznej metody MDS

Klasyczna wersja metody MDS polega na znalezieniu takiej macierzy pozycji punktów Y , dla której odległości euklidesowe między tymi punktami są możliwe najbliższe wartościom z zadanej symetrycznej macierzy odległości D . Działanie algorytmu opiera się na fakcie, że macierz współrzędnych X może pochodzić z dekompozycji macierzy B poprzez EVD i jest równa macierzy iloczynów skalarnych $B=XX^T$. Zależność można przedstawić za pomocą wzoru i nazywana jest podwójnym centrowaniem (ang. double centering):

$$B = -\frac{1}{2}JD^{(2)}J$$

gdzie

$$J = (I_n - \frac{1}{n}1_{nn})$$

Z kolei $D^{(2)}$ oznacza podniesienie do potęgi drugiej każdego elementu z macierzy odległości D . Macierze I_n oraz 1_{nn} są macierzami o wymiarach $n \times n$. Macierz I_n jest macierzą jednostkową (lub inaczej nazywaną identycznościową), która zawiera same jedynki na głównej przekątnej macierzy, zaś pozostałe wartości równe są zero. Z kolei macierz 1_{nn} zawiera same jedynki. Macierz B powinna być symetryczna i nieujemnie określona, o ile podana macierz odległości D odpowiada danej konfiguracji punktów w przestrzeni euklidesowej.

Następnym krokiem, który pozostał jest dekompozycja macierzy B za pomocą EVD:

$$B = K L K^T$$

gdzie kolumny macierzy K są kolejnymi wektorami własnymi macierzy B , a macierz L jest macierzą diagonalną (wszystkie współczynniki leżące poza główną przekątną są zerowe) z wartościami własnymi macierzy B . Tym samym przechodząc do sedna algorytmu, aby zmniejszyć ilość wymiarów do dwóch należy przyjąć w następnych obliczeniach pierwiastek kwadratowy z wartości macierzy L oraz odpowiadających im wektory z macierzy K :

$$Y = K_m \sqrt{L_m}$$

gdzie Y jest szukaną macierzą współrzędnych obiektów, z kolei L_m jest macierzą diagonalną o rozmiarze m z największymi wartościami własnymi (ang. eigenvalues) oraz macierzą K_m o rozmiarze m zawierającą odpowiadające wektory własne (ang. eigenvectors).

W przypadku, gdy po dekompozycji macierzy B występuje więcej ujemnych wartości własnych niż ilość wymiarów, do której chcemy zredukować macierz technika skalowania klasycznego jest niewystarczająca.

2.4.1 Przykład obliczeń

Podrozdział ten ma za zadanie zastosować algorytm MDS na przykładzie danych podobieństwa między czterema pierwszymi państwami z badanego zbioru



w rozdziale 4.2. W tym przypadku wejściowa macierz odległości wygląda następująco:

Tabela 2.1 Przykładowa macierz odległości

	Brazil	Congo	Cuba	Egypt
Brazil	0	4.83	5.28	3.44
Congo	4.83	0	4.56	5.00
Cuba	5.28	4.56	0	5.17
Egypt	3.44	5.00	5.17	0

Pierwszym krokiem jest podniesienie każdego elementu macierzy wejściowej do potęgi drugiej:

$$D^{(2)} = \begin{bmatrix} 0 & 23.33 & 27.88 & 11.83 \\ 23.33 & 0 & 20.79 & 25 \\ 27.88 & 20.79 & 0 & 26.73 \\ 11.83 & 25 & 26.73 & 0 \end{bmatrix}$$

Następnym etapem jest policzenie macierzy J . Skoro macierz odległości ma wymiary 4×4 to $n=4$. Stąd:

$$J = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

W ten sposób podstawiamy J do wzoru na obliczenie macierzy B :

$$B = -\frac{1}{2} J D^{(2)} J = \begin{bmatrix} 7.29 & -3.62 & -5.11 & 1.44 \\ -3.62 & 8.81 & -0.8 & -4.39 \\ -5.11 & -0.8 & 10.38 & -4.47 \\ 1.44 & -4.39 & -4.47 & 7.42 \end{bmatrix}$$

Mając tak przygotowaną macierz B można przejść do dekompozycji EVD.

Wynika z niej, że $B = Q \Lambda Q^T$, stąd:

$$Q = \begin{bmatrix} 0.5 & -0.5 & -0.71 & -0.04 \\ -0.39 & -0.5 & 0.12 & -0.76 \\ -0.6 & -0.5 & -0.1 & 0.62 \\ 0.49 & -0.5 & 0.69 & 0.19 \end{bmatrix} \text{ oraz } \Lambda = \begin{bmatrix} 17.74 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 5.8 & 0 \\ 0 & 0 & 0 & 10.34 \end{bmatrix}$$

Jak już wspomniano na wstępie pracy, dane wielowymiarowe są skalowane do dwóch wymiarów, dlatego trzeba wziąć pod uwagę dwie największe wartości λ (ang. eigenvalues) z macierzy Λ oraz odpowiadające im wektory

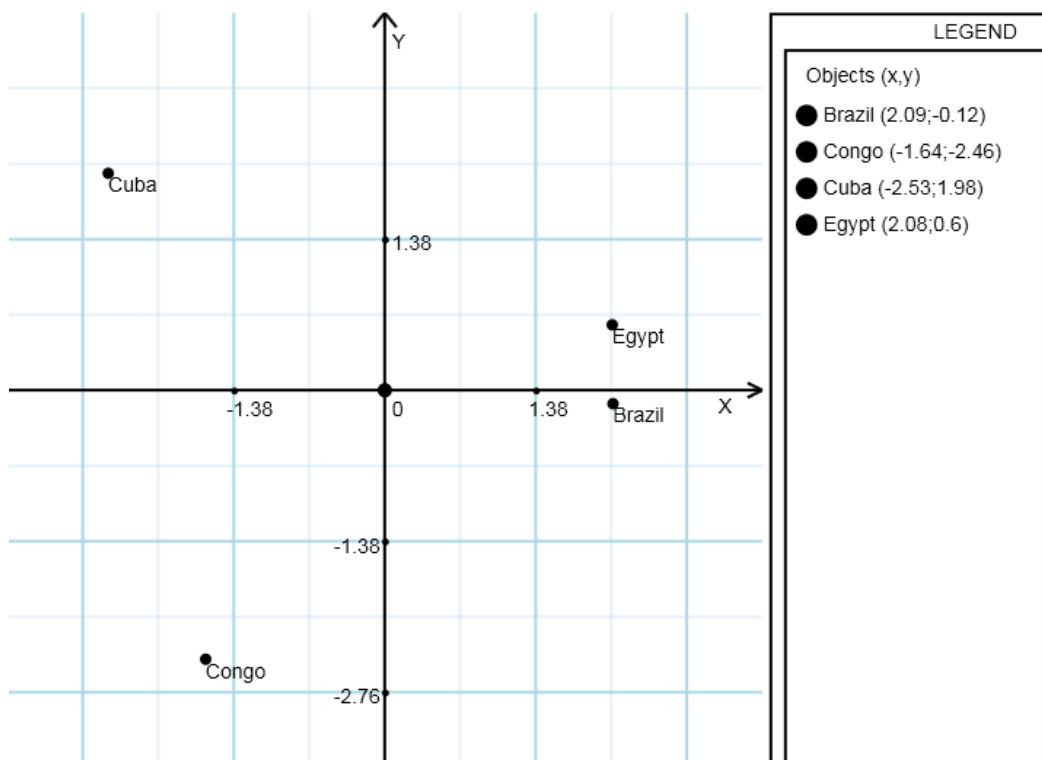
ε (ang. eigenvectors) powiązane tym samym indeksem kolumny. Znalezione

$$\text{elementy wynoszą kolejno: } \lambda_1 = 17.74, \lambda_2 = 10.34, \varepsilon_1 = \begin{pmatrix} 0.5 \\ -0.39 \\ -0.6 \\ 0.49 \end{pmatrix}, \varepsilon_2 = \begin{pmatrix} -0.04 \\ -0.76 \\ 0.62 \\ 0.19 \end{pmatrix}$$

Podsumowując, szukana konfiguracja współrzędnych X dla macierzy wejściowej jest obliczana poprzez wymnożenie wektorów ε oraz wartości λ , z których trzeba najpierw obliczyć pierwiastek kwadratowy:

$$X = \begin{bmatrix} 0.5 & -0.04 \\ -0.39 & -0.76 \\ 0.6 & 0.62 \\ 0.49 & 0.19 \end{bmatrix} \begin{bmatrix} \sqrt{17.74} & 0 \\ 0 & \sqrt{10.34} \end{bmatrix} = \begin{bmatrix} 2.09 & -0.12 \\ -1.64 & -2.46 \\ -2.53 & 1.98 \\ 2.08 & 0.6 \end{bmatrix}$$

co z kolei zostało zaprezentowane w postaci wizualizacji na rysunku 2.1.



Rysunek 2.1 Reprezentacja graficzna szukanego wyniku metody MDS

2.5 Obliczanie błędów skalowania

Po znalezieniu rozwiązania metodą MDS należy zmierzyć jak bardzo ona odpowiada wstępnym założeniom. Model MDS wymaga, aby każdą zaobserwowaną wartość bliskości zmapować dokładnie, do odpowiadającemu mu dystansu. Niemniej jednak, dane empiryczne zawsze zawierają pewien zakres

szumu w związku z nieprecyzyjnym pomiarem czy niesolidnością. Pomimo tego, nawet jeśli dane zawierają błędy to przybliżona reprezentacja tworzy jeszcze lepsze wizualizacje (wydajniejsze, solidniejsze i bardziej odpowiadające rzeczywistości) w porównaniu do tych formalnie idealnych, które pozbywają się zakłóceń.

Jeżeli badacz posiada teorię, którą chce zbadać i w tym celu tworzy macierz odległości, to z kolei inny będzie zainteresowany w sprawdzeniu jak bardzo ta teoria jest w stanie wyjaśnić pochodzące dane. Dlatego poszukiwana jest najlepsza reprezentacja przez metodę MDS. Co więcej, jeśli wynik będzie posiadał zbyt dużo błędów to ktokolwiek może obalić wysuniętą teorię. Oczywiście wcześniej zapoznając się z tym, w jaki sposób taka teoria wpisuje się w dane wejściowe. Każda reprezentacja, która jest na tyle precyzyjna do sprawdzenia tej teorii jest wystarczająco dokładna. [BORG 2005]. Stąd pomysł na aplikację, która będzie w stanie ukazać błąd wynikający z redukcji danych metodą MDS z uwzględnieniem każdej pary obiektów.

Z kolei, jeśli chodzi o najbardziej oczywiste obliczenie błędu to nasuwa się wzór na sumę kwadratów:

$$\sum_{(i,j)} (D_{ij} - DX_{ij})^2$$

gdzie i, j oznacza kolumnę i wiersz danej macierzy, D oznacza macierz wejściową dystansów, a DX macierz wyjściową dystansów powstałą ze znalezionych współrzędnych. Jednakże, ten wzór nie oddaje pełni informacji i prowadzi do błędnych interpretacji wyniku. Duże wartości zawarte w macierzy niekoniecznie wskazują na słaby wynik. Aby tego uniknąć należy znormalizować wartość błędu dzieląc przez sumę kwadratu elementów macierzy wejściowej D , w ten sposób wzór wygląda następująco:

$$\sqrt{\frac{\sum_{(i,j)} (D_{ij} - DX_{ij})^2}{\sum_{(i,j)} D_{ij}^{(2)}}}$$

Funkcja ta jest nazywana często funkcją kosztu (ang. stress), czasami znormalizowaną funkcją kosztu (ang. normalised stress) lub Stress-1 [KRUS 1964]. Oczywiście dąży ona do zera. Kiedy jest równa zero to macierz wejściowa D jest równa wyjściowej DX .



W aplikacji zawarto bardziej czytelny opis znormalizowanego błędu według poniższej tabeli, która została zasugerowana w pracy [KRUS 1964].

Tabela 2.2 Wartość funkcji kosztu oraz odpowiadające mu oznaczenie

Znormalizowany błąd	Poprawność znalezionej rozwiązania
większy niż 0.20	słabe
większy niż 0.10	poprawne
większy niż 0.05	dobrze
większy niż 0.025	bardzo dobrze
większy niż 0.00 i mniejszy niż 0.025	idealne

3. Projekt systemu wizualizacji danych wielowymiarowych

Rozdział ten opisuje szczegółowo system wizualizacji danych wielowymiarowych wykorzystujący metodę skalowania wielowymiarowego (MDS) i uwzględniający niedokładności w generowanych położeniach obiektów przy użyciu odległości nieeuklidesowych. Jest on niejako kontynuacją rozdziału 2.5, w którym opisywane są sposoby liczenia błędu powstałego w wyniku skalowania. Jednakże obecnie aplikacje jedynie prezentują pojedynczy błąd poprzez wyświetlanie (Rysunek 3.2) lub wykres z kolejnymi etapami zmiany tej wartości. Nikt do tej pory nie pokazał tego, co kryje się pod tą liczbą lub też z czego dokładnie ona się składa (Rysunek 3.1).

DIFFERENCES											
#	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	
o1	0	0.63	2.32	4.09	0.79	1.48	0.45	1.41	0.53	2.5	
o2	0.63	0	2.39	0.64	2.77	0.12	0.63	0.78	0.08	0.96	
o3	2.32	2.39	0	5.9	4.07	3.2	2.49	3.78	2.01	5.94	
o4	4.09	0.64	5.9	0	0.95	-0.06	1.21	0.91	0.62	2.09	
o5	0.79	2.77	4.07	0.95	0	0.75	-0.39	0.02	-0.07	-1.51	
o6	1.48	0.12	3.2	-0.06	0.75	0	1.28	3.02	-1.67	-0.04	
o7	0.45	0.63	2.49	1.21	-0.39	1.28	0	-1.46	2.95	0.44	
o8	1.41	0.78	3.78	0.91	0.02	3.02	-1.46	0	1.19	-0.47	
o9	0.53	0.08	2.01	0.62	-0.07	-1.67	2.95	1.19	0	0.76	
o10	2.5	0.96	5.94	2.09	-1.51	-0.04	0.44	-0.47	0.76	0	

Rysunek 3.1 Różnice dystansów między poszczególnymi obiektami w wyniku MDS

Kruskal stress is poor (0.4225). Found min(-1.67), max(5.94) differences

Rysunek 3.2 Ogólna wartość błędu określająca jakość rozwiązania MDS

Obecnie wszystkie aplikacje zatrzymały się na rysowaniu samych obiektów (Rysunek 2.1). Skoro MDS przeprowadza się w celu wizualizacji jak najlepszej reprezentacji danych wielowymiarowych to powinna ona uwzględniać poszczególne różnice, które istnieją między obiektami po zmniejszeniu wymiarów. W ten sposób można precyzyjniej określić, ile tak naprawdę wynosi błąd w poszczególnych relacjach między obiektami, a nie tylko jako



znormalizowaną sumę różnic, która nie mówi nic poza ogólnym stwierdzeniem jaka jest jakość z zastosowania metody MDS. Dlatego też system potrafi dodatkowo zaprezentować za pomocą różnego typu linii w przestrzeni nieeuklidesowej poprawne odległości z macierzy wejściowej.

Co więcej, aby dobrze rozróżnić czy też pogrupować linie, aplikacja oferuje do włączenia kolorowanie według różnicy oryginalnego dystansu z macierzy wejściowej, a długości obliczonej z macierzy wyjściowej konfiguracji współrzędnych. W ten sposób bardzo szybko można podjąć decyzję o wyborze prawidłowych obiektów. Dodatkowo, wizualizacja interaktywnie reaguje na zaznaczenie linii lub obiekty wygaszając pozostałe.

3.1 Analiza zapotrzebowań

System musi posiadać intuicyjny i spójny interfejs, który będzie oferował w czytelny i łatwy sposób dostęp do poszczególnych modułów i opcji aplikacji. Dodatkowo powinien umożliwiać w module dane:

- poprawne ładowanie danych oraz nagłówków w formacie csv wraz z usuwaniem dodatkowych spacji. Dane mogą być:
 - obiektami (w postaci współrzędnych 0.1,3.6 2,1 itd.),
 - macierzą bliskości (w postaci symetrycznej macierzy),
- generowanie macierzy odległości na podstawie wygenerowanych obiektów z przestrzeni o współrzędnych:
 - dodatnich,
 - ujemnych,
 - mieszanych (dodatnich i ujemnych po połowie),
- wybór znaku rozdzielającego ładowanie danych oraz ich nagłówki,
- ustawienie maksymalnego wymiaru dla ładowanych lub generowanych danych,
- wyświetlanie załadowanej macierzy dystansów,
- oznaczenie opcji do automatycznych obliczeń po załadowaniu danych, które umożliwia przejście od razu do wizualizacji,
- samodzielne obliczanie kolejnych kroków algorytmu MDS,
- edycję wiersza danych,
- walidację ładowanych macierzy dystansów,
- ustawienie wybranego koloru dla danego obiektu,



- wyłączenie i włączenie konkretnego obiektu.

Na następnej stronie „Decomposition” w module dekompozycji EVD musi spełniać założenia takie jak:

- obliczanie współrzędnych do wizualizacji ,
- sprawdzanie czy jest wymagana ilość nieujemnych wartości z EVD oraz wyświetlenie odpowiedniej informacji w przypadku, gdy nie został spełniony ten warunek.

Po poprawnych obliczeniach system na stronie „Output”:

- wyświetla obliczone współrzędne obiektów,
- umożliwia obliczenie poszczególnych różnic między obiektami na podstawie macierzy wejściowej dystansów oraz macierzy wyjściowej dystansów policzonej z wyświetlanych współrzędnych.

Ostatnią stroną jest „Drawing”, która odpowiada za:

- rysowanie obiektów według obliczonych współrzędnych,
- rysowanie linii między obiektami na podstawie różnic:
 - przerywanych dla różnic ujemnych,
 - prostych dla różnic zerowych,
 - falistych dla różnic pozytywnych,
- kolorowanie linii według mapy kolorów „Jet” bez względu na wartości
 - możliwość włączenia lub wyłączenia (domyślnie wyłączone),
 - możliwość włączenia efektu „relatively”, który odpowiada za wygenerowanie mapy kolorów relatywnie do wartości,
 - odcienie niebieskie do zielonego reprezentują ujemne wartości,
 - odcienie zielonego reprezentują okolice wartości zero,
 - odcienie od zielonego do czerwonego reprezentują pozytywne wartości,
- włączenie lub wyłączenie aspektu dla osi układu współrzędnych,
- włączenie lub wyłączenie rysowania linii pomocniczych,
- określenie jak często mają rysować się pomocnicze linie oraz wartości na osiach współrzędnych,
- określenie typu algorytmu dla linii przerywanych:



- minimum (rysuje przerywane linie na podstawie najmniejszej wartości bezwzględnej z ujemnych różnic),
 - percentage (rysuje procentowo, czyli na podstawie zadanej długości daną ilość linii),
 - określenie ilości linii do rysowania,
- przesuwanie wykresu do góry, do dołu, w lewo i prawo,
- zwiększanie i zmniejszanie powiększania wizualizacji,
- zatrzymanie w dowolnym momencie rysowania (asynchroniczne rysowanie),
- zapis wizualizacji wraz z legendą:
 - do pliku graficznego png,
 - wektorowo do pliku xps.

3.2 Wymagania pozafunkcjonalne

Aplikacja została zaprojektowana i zaimplementowana w oparciu o najnowsze technologie desktop'owe [DESKTOP]. Napisana jest w języku C# (rozdział 3.3.2) w technologii WPF (rozdział 3.3.1) w systemie Windows XP SP3 32 bit [WINXP] z wykorzystaniem systemu kontroli wersji (ang. subversion, SVN) [SVN] o nazwie TortoiseSVN [TSVN], aplikacji internetowej assembla [ASSEMBLA], służącej do zarządzania projektem oraz systemu moqups [MOQUPS] do rysowania makiet (szkiców interfejsów). Środowisko takie zapewniało wydajne, stabilne i przejrzyste zarządzanie projektem. Dodatkowo wykorzystano aplikację Microsoft Visio 2010 [VISIO] do budowania diagramów klas. System jest uruchomiony na sprzęcie o następujących parametrach:

- procesor Intel Core 2 Duo CPU T5250 1,50 GHZ,
- pamięć RAM 3,00 GB.

System korzysta podczas rysowania wizualizacji z asynchronicznych operacji, które umożliwiają podgląda aktualnego stanu procesu oraz zatrzymanie go w dowolnej chwili. Dodatkowo system posiada obsługę wyjątków, które są zapisywane do pliku logu zachowując przy tym wysoką stabilność.

Projekt został zrealizowany w oparciu o wzorzec architektoniczny Model-Widok-Kontroler (ang. Model-View-Controller, MVC) [MVC]. Dzięki



wykorzystaniu technologii WPF aplikacja jest skalowalna wektorowo na różnych rozdzielczościach ekranu.

3.3 Opis technologii, narzędzi i bibliotek

Rozdział ten został poświęcony na opis wszystkich najważniejszych narzędzi oraz używanych bibliotek w systemie. Z należytą starannością zostały one wybrane, aby wzbogacić system zachowując przy tym prostotę i ograniczając zbędne dodatki.

3.3.1 Technologia WPF

Windows Presentation Foundation (WPF) [WPF] jest nową technologią budowania aplikacji w systemie Windows. Zamiast opierania się na starym interfejsie do tworzenia aplikacji (ang. Graphics Device Interface, GDI) [GDI] używa bezpośrednio DirectX. WPF dostarcza jednolitej metodyki budowania aplikacji separując interfejs graficzny od logiki biznesowej. Tworzenie widoków jest oparte na języku XML [XML], a precyzyjniej nazywa się on XAML (ang. Extensible Application Markup Language). WPF integruje interfejs użytkownika, grafikę 2D i 3D, multimedia, dokumenty oraz generowanie i rozpoznawanie mowy. Graficzny interfejs użytkownika wykorzystuje grafikę wektorową, budowaną z użyciem akceleratorów grafiki 3D i efektów graficznych, które zastąpiły dotychczasowe biblioteki DirectX [DIRECTX].

3.3.2 Język C#

Język C# [CSHARP] jest połączeniem języków programowania Object Pascal [PASC], Delphi [DELP], C++ [C++] i Java [JAVA]. Główne jego zalety:

- obiektowość hierarchiczna – tylko jeden element nadrzędny,
- odśmiecanie pamięci – zarządzanie pamięcią,
- właściwości – dodatkowe elementy klas,
- delegaty, zdarzenia – odpowiedniki wskaźników [WSKA],
- refleksje i atrybuty klas – umożliwia analizę struktury kodu,
- typy ogólne – mechanizm podobny do szablonów w C++,
- dynamiczne tworzenie kodu,
- bogata biblioteka klas – umożliwia rozwój wielu aplikacji.



3.3.3 Narzędzie Syncfusion Metro Studio

Syncfusion Metro Studio [SYNC] jest aplikacją, która za darmo udostępnia zbiór ponad dwóch i pół tysiąca wektorowych ikon w stylu Metro (więcej o tym w rozdziale 3.3.8). Dzięki niej system zawiera elementy graficzne na przyciskach, które są skalowalne.

3.3.4 Narzędzie Inkscape

Inkscape [INK] jest darmowym narzędziem do tworzenia grafiki wektorowej. Zawiera elastyczne narzędzia do tworzenia tekstów, krzywych i innych kształtów. Posiada też opcję importowania powstałej grafiki wektorowej do formatu ciągu liczb. Dzięki czemu można w łatwy sposób przenieść grafikę do aplikacji WPF. W ten sposób powstało logo systemu.

3.3.5 Narzędzie IcoFX

IcoFx [ICOFX] jest narzędziem do tworzenia ikon w różnych formatach oraz rozdzielczościach, dzięki niemu grafika jest wysokiej jakości. Program otrzymał liczne wyróżnienia za profesjonalność aplikacji.

3.3.6 Narzędzie Wolfram|Alpha

Wolfram|Alpha [WOLF] jest innowacyjnym projektem, którego podstawową ideą jest przekazywanie wiedzy nie przez wyszukiwanie informacji, ale przez dynamiczne obliczenia i algorytmy. Posiada bazę wiedzy na temat elementarnych modeli, metod i rozwiązań, aby obliczyć każdą formułę. Narzędzie internetowe pomogło zrozumieć ideę obliczania długości sinusoid.

3.3.7 Narzędzie Microsoft Visual C# 2010 Professional

Microsoft Visual C# 2010 Professional [VS] to środowisko zintegrowane zawierające zestaw narzędzi programistycznych pozwalający na tworzenie aplikacji sieciowych, usług sieciowych i serwisów internetowych. Przeznaczone do tworzenia bibliotek, klas, aplikacji konsolowych i okienkowych na platformie .NET Framework 4 [.NET] w języku C# (rozdział 3.3.2).



3.3.8 Biblioteka MahApps.Metro

MahApps.Metro jest projektem tworzonym przez społeczność w celu dostarczenia prostego stylu wzorowanego na stylistyce Modern UI (wcześniej znany jako Modern User Interface lub Metro UI). Interfejs taki charakteryzuje się przejrzystymi napisami, minimalizmem i kontrastową kolorystyką. Co więcej, oprócz stylu dostarcza także niestandardowe kontrolki pod aplikacje WPF [METRO].

3.3.9 Biblioteka DotNumerics

DotNumerics [DOTN] jest biblioteką przeznaczoną do numerycznych obliczeń. Zawiera zaawansowane algorytmy, w tym dla algebry liniowej bazującej na translacji kodu m.in LAPACK'a (Linear Algebra PACKage) [LAPACK] z języka Fortran [FORTRAN] do C#. Biblioteka ta służy w systemie do rozkładu EVD.

3.3.10 Biblioteka NLog

NLog jest darmowym projektem służącym do logowania na platformie .Net z zaawansowanymi opcjami logowania i zarządzania. Dzięki niemu w łatwy sposób można utrzymywać wysokiej jakości logi bez względu na rozmiar czy złożoność [NLOG].

3.4 Opis struktury systemu

Diagramy klas przedstawiają strukturę statyczną systemu i składają się z obiektów klas między innymi występujących w diagramach przypadków użycia czyli pokazują z czego składa się system. Klasy opisują różne rodzaje obiektów występujących w systemie, zaś diagramy klas prezentują zależności występujące pomiędzy nimi. Diagramy klas znajdują się w dodatkach w rozdziale 7.3.

Z uwagi na szczegółowość diagramów klas opisano najważniejsze obiekty:

- StatusObject – klasa używana do obsługi statusu w aplikacji,
- BaseModel – abstrakcyjna klasa będąca bazą dla modeli danych,
- DecompositionModel – model danych dla strony „Decomposition”,
- MainControllerModel – główny model danych dla kontrolera,
- MatrixColumn – model dla kolumny widoku macierzy,
- DataModel – model danych dla strony „Data”,



- MatrixRow – model dla wierszy widoku macierzy,
- OutputModel – model danych dla strony „Output”,
- DrawingModel – model danych dla strony „Drawing”,
- GridListModel – model danych dla widoku macierzy,
- CanvasModel – główny model danych obsługujący wizualizację,
- BasePage – główna bazowa klasa dla stron,
- DrawingPage – klasa odpowiadająca za widok strony „Drawing”,
- MainPage – klasa odpowiadająca za główny widok,
- OutputPage – klasa odpowiadająca za widok „Output”,
- DecompositionPage – klasa odpowiadająca za widok „Decomposition”,
- DataPage – klasa odpowiadająca za widok „Data”,
- FunctionBase – klasa bazowa dla funkcji liniowych i nieliniowych,
- FitBroken – klasa odpowiadająca za linie przerywane,
- FitSin – klasa odpowiadająca za linie krzywe tzw. sinusoidy,
- Sin – klasa odpowiada za liczenie funkcji sinus,
- FitStraight – klasa odpowiada za funkcje prostych linii,
- Paths – klasa statyczna zawierająca kształty dla wektorów,
- MetroWindow – klasa bazowa odpowiada za widok okien,
- MainWindow – klasa odpowiada za główne okno aplikacji,
- MainController – klasa kontroler odpowiada za główną logikę aplikacji,
- ClickableLabel – klasa odpowiada za element menu w formie napisów,
- CanvasViewExt – rozszerzona klasa odpowiada za rysowanie,
- ListViewExt – klasa odpowiada za widok macierzy,
- ColorMap – klasa implementuje mapę kolorów w standardzie „Jet”,
- ClassicMds – klasa implementuje klasyczny MDS,
- Evd – klasa implementuje rozkład EVD,
- ExtPoint – rozszerzona klasa implementuje dane punktów,
- DataSource – klasa zawierająca implementację ładowania danych,
- ScaleHelper – klasa odpowiada za skalowanie podczas rysowania,
- DisposedObject – klasa bazowa implementuje uwalnianie zasobów.

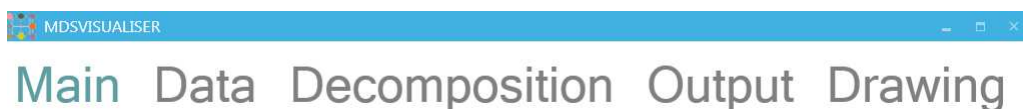
Większość klas dziedziczy po obiekcie klasy „DisposedObject”, który implementuje interfejs IDisposable służący do zwalniania zasobów. W ten

sposób system zachowuje się stabilnie i nie obciąża jednostki roboczej. Z uwagi na brak wielokrotnego dziedziczenia w języku C# klasa bazowa „BasePage” dodatkowo implementuje ten sam interfejs IDisposable. Dziedziczą po niej wszystkie strony, aby powiadamiać kontroler o akcjach. Z kolei abstrakcyjna klasa bazowa „BaseModel” implementuje interfejs „INotifyPropertyChanged” służący do powiadamiania stron o zmianie wartości w modelu. W ten sposób strony xaml w WPF asynchronicznie reagują na zmiany w modelu poprzez odpowiednie odwołanie. Dodatkowo „BaseModel” dziedziczy po klasie „StatusObject”, aby mieć możliwość powiadamiania o błędach oraz innych informacji. Jest podstawą dla wszystkich modeli, które są przetwarzane przez strony i obsługiwane przez kontroler. Oczywiście „StatusObject” dziedziczy z kolei po „DisposedObject” opisanym wyżej.

3.5 Opis implementacji

System został opracowany w formie kreatora (ang. wizard), aby ujednolicić widoki i uprościć interfejs. Projekt graficzny „Mds.UI” składa się z poniższych widoków i odpowiadających im modułów:

- MainPage – główny i pierwszy widok aplikacji,
- DataPage – widok obsługi danych,
- DecompositionPage – widok wyniku EVD,
- OutputPage – widok wynikowych współrzędnych oraz błędu skalowania,
- DrawingPage – widok odpowiadający wizualizacji metody MDS.

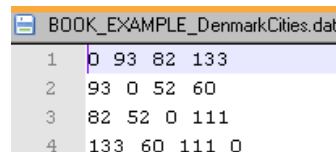


Rysunek 3.3 Widok menu aplikacji

Aplikacja bazuje na architekturze Model-Widok-Kontroler. Za kontroler odpowiada klasa „MainController”, który posiada implementację modelu „MainControllerModel”. Jest to jądro systemu, które zarządza całym systemem. Zawiera referencje do wszystkich stron i ich modeli oraz do klasy „ClassicMds”, która realizuje algorytm metody MDS. Instancja kontrolera bazuje w głównym widoku klasy „MainWindow”, która to jest podstawą interfejsu graficznego całego systemu. Zawiera ona w nagłówku menu składające się z elementów klasy „ClickableLabel”, które są rozszerzeniem standardowych napisów. Reagują na

klikanie i odpowiednio wtedy wyróżniają się kolorystyką w celu zachowania minimalizmu i estetyki. Ponadto w stopce głównego okna jest zawarty opis aktualnego statusu oraz przyciski „back” (wstecz) oraz „next” (dalej) służące do nawigacji po systemie. Tekst w statusie informuje użytkownika na każdym kroku o ewentualnych błędach i możliwych akcjach. Napis na przycisku „next” może zostać zamieniony na „draw” (rysuj wynik metody MDS) w przypadku wybrania opcji „automatic” (wykonaj obliczenia algorytmu MDS automatycznie).

Przechodząc do widoku ładowania danych możemy dowolnie ładować dane pochodzące z macierzy dystansów w kształcie macierzy symetrycznej widocznej na rysunku 3.4 poniżej. Implementuje to klasa „DataSource” z projektu „Mds.Core”. Co więcej, można wcześniej zadeklarować maksymalną wartość wymiarów do załadowania lub wygenerowania.



	0	93	82	133
1	0	93	82	133
2	93	0	52	60
3	82	52	0	111
4	133	60	111	0

Rysunek 3.4 Przykład macierzy symetrycznej

Dodatkowo można także załadować współrzędne obiektów w postaci widocznej poniżej na rysunku 3.5.



1	0.1,3.6	2,1	4.2,5.1	6,9	-2,-4	-2.3,-4.1
---	---------	-----	---------	-----	-------	-----------

Rysunek 3.5 Przykład współrzędnych do załadowania

Dane przyjmują format CSV i odpowiednio dokonują rozdzielenia danych na podstawie wybranego znaku z rozwijalnej listy. Co więcej system usuwa dodatkowe spacje. Potrafi także wygenerować obliczone dystanse na podstawie wybranych losowo współrzędnych obiektów. Podczas ładowania danych obliczana jest maksymalna wartość dystansu na podstawie której system skaluje długości. Jeżeli wartość mieści się między zero i dwadzieścia pięć to wartość skalowalna wynosi jeden, w przeciwnym wypadku system odpowiednio zwiększa ją mnożąc przez dziesięć. Celem jest przyspieszenie operacji w systemie, szczególnie rysowania.

Po załadowaniu danych otrzymujemy informacje z systemu w stopce o tym, czy dane zostały poprawnie załadowane. Jeżeli tak to opcja „automatic” jest dostępna do zaznaczenia. Umożliwia ona pominięcie ręcznego obliczania



progowych danych i przejście od razu do widoku wizualizacji na podstawie wyniku metody MDS. W przypadku ręcznego wyboru należy obliczyć najpierw macierz B poprzez akcję „MAKE B MATRIX”, a następnie przeprowadzić jej dekompozycję poprzez algorytm EVD, którego implementacja znajduje się w klasie „EVD” w projekcie „Mds.Core”. Po wywołaniu akcji „RUN EVD” system automatycznie przechodzi do strony „Decomposition”. W tym momencie ukazuje się dekompozycja macierzy poprzez EVD, która jest wyświetlana odpowiednio na dwie listy „EIGEN VALUES” oraz „EIGEN VECTORS”. O ile oczywiście algorytm odnalazł co najmniej dwie nieujemne wartości własne macierzy B (dwie, ponieważ system skaluje wielowymiarowo do dwóch wymiarów jak zaznaczono to na wstępie). W przeciwnym wypadku zostanie wyświetlona informacja w stopce o ponownym załadowaniu poprawnych danych i zablokowana możliwość przejścia dalej oraz obliczenia współrzędnych.

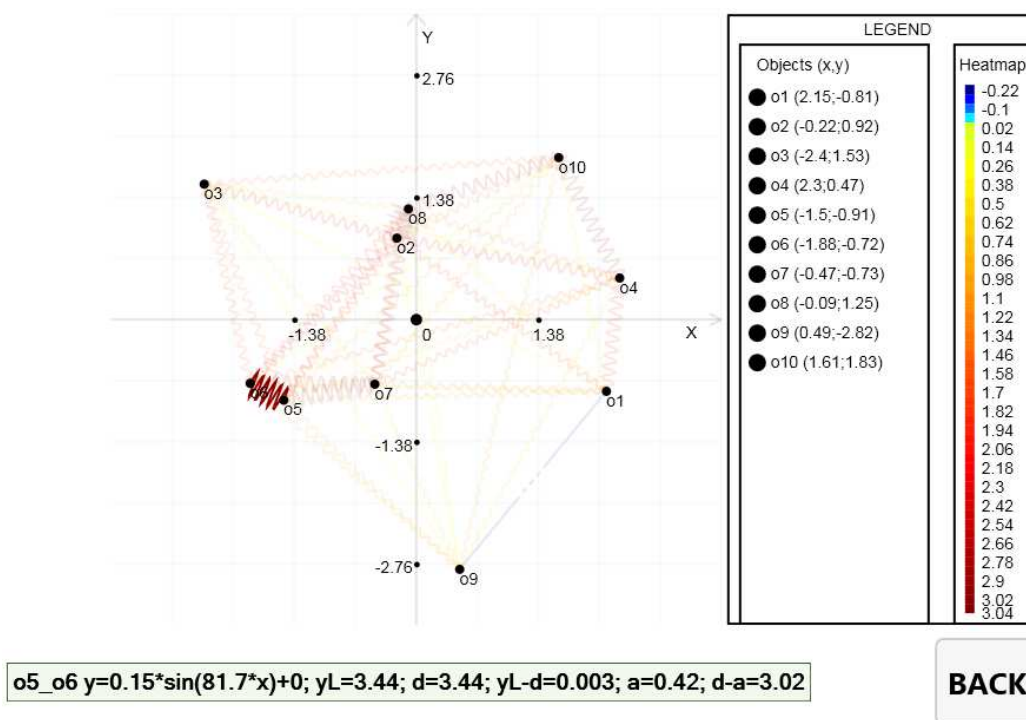
Jednakże zakładając dalsze poprawne działanie użytkownika możemy obliczyć współrzędne poprzez akcję „CALCULATE COORDINATES”, która to automatycznie przeniesie na stronę „Output” i zaprezentuje wynikowe współrzędne do zaprezentowania. Pozostał ostatni krok „CALCULATE DIFFERENCES”, który porówna odległości między macierzą wejściową, a odległościami obliczonymi z wynikowej konfiguracji współrzędnych. W ten sposób system wyświetli poszczególne różnice między obiektami oraz obliczy na końcu podsumowujący błąd tzw. „Kruskal stress”. Jak zostało to wspomniane w rozdziale dotyczącym obliczeń błędów aplikacja ukazuje także bardziej przyjazną formę tej wartości poprzez interpretację słowną.

Ostatnim modulem jest „Drawing”, do którego przechodzimy po wywołaniu akcji „DRAW”. Jeżeli wybraliśmy opcję automatycznych obliczeń to system od razu wyświetli tą akcję. Po jej wywołaniu zostanie wyświetlona wizualizacja w wyniku metody MDS, która jest realizowana poprzez klasę „ClassicMds” w projekcie „Mds.Core”. Oczywiście o ile podczas rozkładu EVD znaleziono odpowiednie wartości. W przeciwnym wypadku użytkownik zostaje cofnięty do poprzedniego kroku, aby mógł załadować poprawne dane.

Moduł rysowania bazuje na rozszerzonej wersji klasy „Canvas” o nazwie „CanvasViewExt”, która zawiera metody do tworzenia potrzebnych figur i linii oraz realizuje zdarzenia klikania. W ten sposób użytkownik po kliknięciu na dany



obiekt lub linie może „wygaszać” pozostałe obiekty oraz wyświetla w ten sposób informacje w stopce aplikacji w formacie: „obiekt źródłowy obiekt docelowy funkcja rysowania; długość tej funkcji (yL); dystans wejściowy (d); różnica między długością wyrysowanej linii, a wejściowym dystansem (yL-d); dystans między wyrysowanymi obiektami (a); różnica między dystansem wejściowym (d), a dystansem wyjściowym (d-a).



Rysunek 3.6 Zaznaczenie linii falistej

Na widoku wizualizacji są oczywiście rysowane obiekty, a także linie reprezentujące prawidłowe odległości z macierzy wejściowej między poszczególnymi obiektami. Obok układu współrzędnych jest rysowana legenda z opisem obiektów oraz typów linii. Standardowo linie kolorowane są tak jak na rysunku 3.16:

- na niebiesko w przypadku różnic ujemnych,
- na zielono w przypadku różnic równych zero,
- na czerwono w przypadku różnic pozytywnych.

Na rysunku 3.6 widać przykład takiej różnicy pozytywnej, o której informacja po zaznaczeniu linii wygląda: $d-a=3.02$.

Nad reprezentacją graficzną widoczny jest pasek postępu informujący o stanie wygenerowanej wizualizacji. Obok niej na prawo i na lewo znajdują się przyciski akcji „OPTIONS” oraz „ACTIONS”. Po ich wywołaniu ukazuje się wysuwane

menu. W ten sposób przy niskiej zajętości obszaru aplikacji umieszczono wiele funkcji pogrupowanych i dostępnych w szybki sposób.

Pierwszą ciekawą opcją dostępną z poziomu „Options” jest włączenie mapy kolorów typu „Jet” dla wygenerowanych linii. Implementacja znajduje się w klasie „Colormap” i jej źródło pochodzi z publikacji [XU 2009]. Generuje ona w podstawowej wersji do 64 kolorów zależnie od wartości minimum i maksimum.

Tabela 3.1 Porównanie mapy kolorów "Jet"

Standardowa mapa kolorów „Jet”	Relatywna mapa kolorów „Jet”

Po włączeniu opcji „relatively” pod opcją „Jet heatmap” algorytm zmienia nieco swoje działanie w ten sposób, aby relatywnie do wartości generował kolory. W wyniku tego wartość zero zawsze ma kolor bliski zielonemu, zaś wartości ujemne pochodzą z zakresu kolorów od zielonego do niebieskiego. Z kolei wartości pozytywne generowane są tylko z zakresu od zielonego do czerwonego.

Oczywiście generowanie linii można wyłączyć (rysunek 2.1) lub włączyć, tak samo jak włączenie zachowania aspektu osi X i Y układu współrzędnych, jednak nie zaleca się tego robić z powodu przekłamania w reprezentowaniu linii. Co więcej można także wyłączyć oraz włączyć generowanie linii pomocniczych, a także manipulować ich ilością oraz częstotliwością pośrednich punktów na osiach w celach informacyjnych.



Na samym końcu opcji wizualizacji znajduje się możliwość wyboru generowania przerywanych linii, o czym więcej w rozdziale 3.5.1.

Przechodząc do sekcji akcji znajdujących się na prawo od grafiki od razu można zauważyć możliwość sterowania przesuwaniem wizualizacji odpowiednimi strzałkami, a także zbliżaniem lub oddalaniem przez oznaczenia lupy. Dodatkowo możemy w każdej chwili przerwać asynchroniczne rysowanie przez wywołanie akcji „STOP”. Z kolei akcja „DRAW” umożliwia reset rysowania. Ostatnią akcją jaką możemy wykonać to zapis całej wizualizacji wraz z legendą do pliku png lub wektorowo do pliku xps (zmodyfikowana metoda na bazie źródła [VUYK 2007]).

Co więcej z racji metodyki tworzenia aplikacji w technologii WPF zostały stworzone liczne konwertery (wszystkie w przestrzeni o nazwie „Mds.UI.Converters”), które odpowiadają za dynamicznie tworzone widoki czy akcje (głównie znikanie lub pojawianie się elementów).

Jeśli zaś chodzi o implementację projektu bazowego „Mds.Core” to warto wspomnieć fakt, że wszystkie klasy odpowiadające za poszczególne linie bazują na abstrakcyjnej klasie bazowej „FunctionBase”. W niej znajduje się między innymi implementacja obliczania Euklidesowego dystansu (pochodzącego z twierdzenia Pitagorasa) między dwoma punktami. Długość między punktem a i b w układzie kartezjańskim wynosi

$$\sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

gdzie n oznacza przestrzeń euklidesową. Dodatkowo w tym pakiecie znajduje się pomocnicza klasa „ScaleHelper”, która odpowiada za:

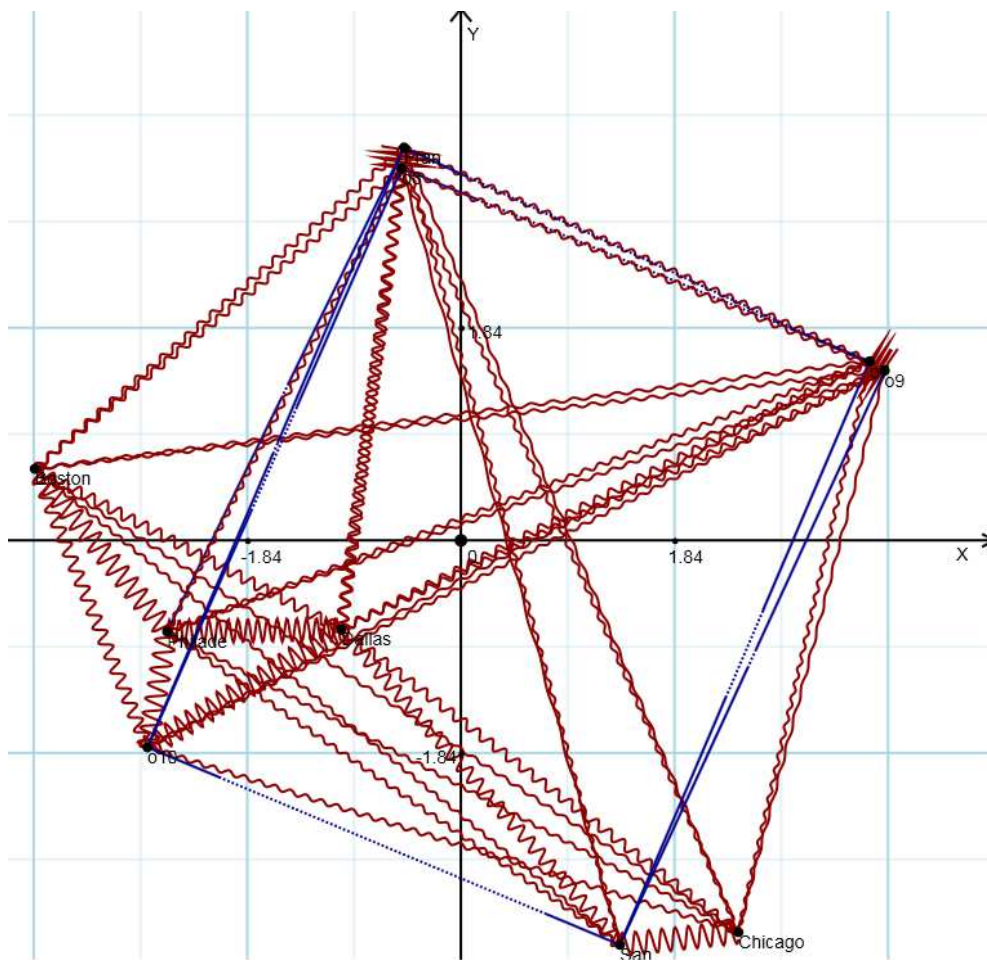
- odpowiednie skalowanie współrzędnych do odpowiedniego rysowania na ekranie,
- generowanie wartości granicznych dla maksymalnych i minimalnych współrzędnych X i Y oraz podczas manipulowania wizualizacją przy operacjach przemieszczania i powiększania oraz pomniejszania.

Oczywiście zawiera też klasę „Extensions”, która rozszerza o nowe metody dostępne klasy.

3.5.1 Generowanie linii przerywanych

Linie przerywane w systemie odzwierciedlają różnice, które są ujemne. Ich długość w macierzy wejściowej jest mniejsza niż długość między wyrysowanymi obiektami. Stąd, żeby zachować prawidłową długość należy uwzględnić przerwy w linii. W systemie są zawarte dwa algorytmy.

Pierwszy z nich o nazwie „minimum” rysuje przerwy w linii na podstawie najmniejszej bezwzględnej wartości z ujemnej różnicy między macierzą wejściową dystansów, a wyjściową. Treść pseudokodu z uwagi na znaczną długość jest zamieszczony w dodatkach w tabeli 7.3. Wynika z niego, że algorytm tworzy symetrycznie przerwy w linii na prawo i na lewo od środka linii. Jeżeli długość do zaprezentowania jest duża to przerwy są dłuższe. Z kolei jeśli przerwy zostały już wyrysowane to algorytm tworzy ciągłe linie od punktu początkowego oraz od końcowego do pierwszej z brzegu przerwy.



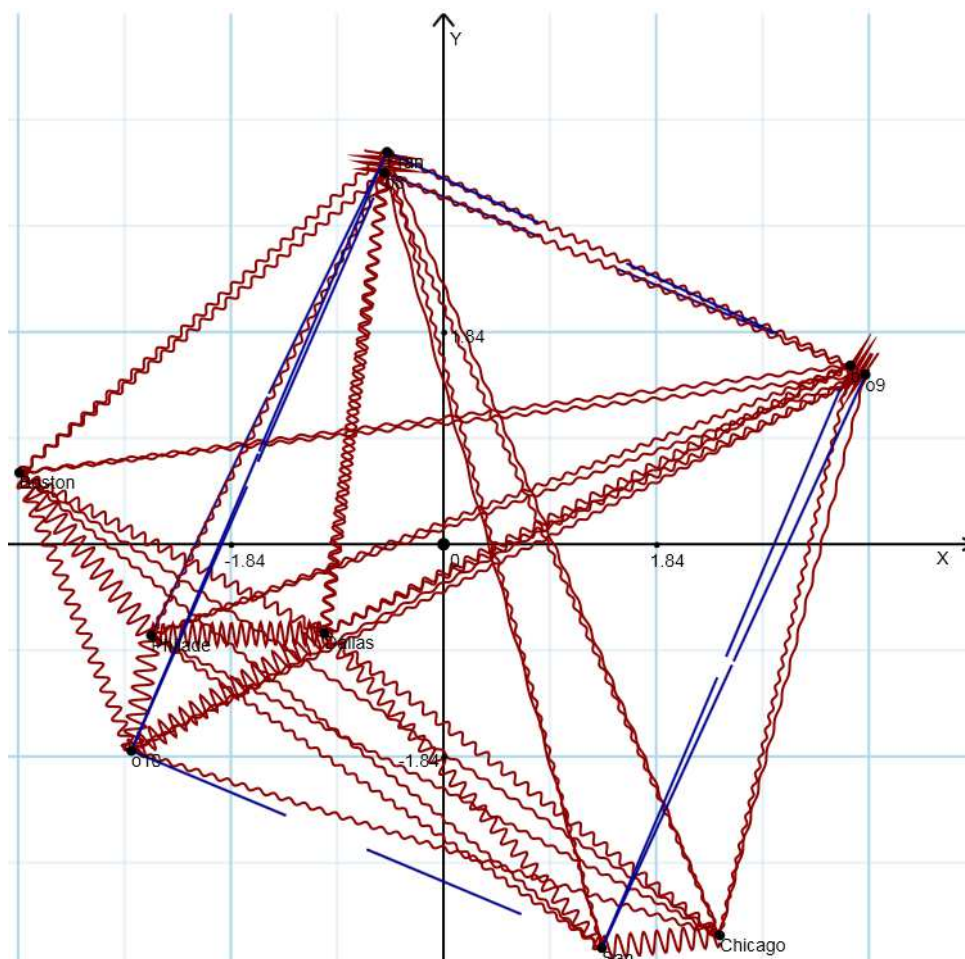
Rysunek 3.7 Przykładowa wizualizacja z włączonym algorytmem „minimum”

Kolejnym algorytmem, który zasugerowano jest dzielenie dystansu między punktami i obliczanie procentowo liczby zaprezentowanych długości na tych partiach. Pseudokod jest następujący:

```

procentDoZaprezentowania = ( ( 100 * oczekiwana_długość ) / długość_międzyobiektami ) / 100;
przesunięcie = długość_międzyobiektami / ilość_kroków;
poprzedniX = 0;
dla (krok=0; krok < ilość_kroków; krok++) {
    DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(poprzedniX));
    długość_częściowej_linii = procentDoZaprezentowania * przesunięcie;
    następnyX = poprzedniX + długość_częściowej_linii;
    znaleziona_długość += długość_częściowej_linii;
    DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(następnyX));
    poprzedniX += przesunięcie;
}

```



Rysunek 3.8 Przykładowa wizualizacja z włączonym algorytmem „percentage”

3.5.2 Generowanie linii falistych o zadanej długości

Linie faliste są tak naprawdę sinusoidami i reprezentują w systemie długości, które w macierzy wejściowej są większe niż w macierzy wyjściowej powstałej

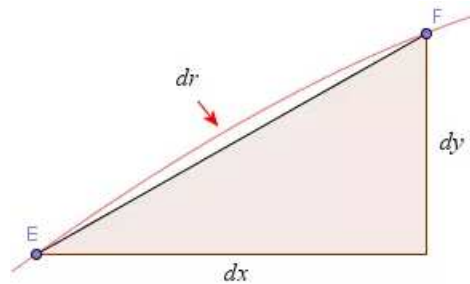
z obliczeń długości między obiektami znalezionymi w wyniku metody MDS. Algorytm bazuje na liczeniu dynamicznym długości. Wynosi ona dla funkcji ciągłej $y=f(x)$ od $x=a$ do $x=b$:

$$\int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx$$

Jeżeli teraz za funkcję y przyjmujemy funkcję sinus:

$$f(x) = A \sin(Fx + P)$$

gdzie A oznacza wysokość fal, F to częstotliwość ich występowania, a P jest przesunięciem, które wynosi zero.



Rysunek 3.9 Częściowa długość fali [BOURNE]

Dodatkowo, jeżeli za zależność dy/dx widoczną na rysunku 3.9 przyjmujemy pochodną funkcji sinus, ponieważ to ona reprezentuje sumę tych krótkich części fali zaznaczonych na rysunku poprzez symbol dr oznaczający:

$$\frac{dy}{dx} = AF \cos(Fx + P)$$

to wzór na długość funkcji sinus obliczona jest na podstawie całki:

$$\int_a^b \sqrt{1 + A^2 F^2 \cos^2(Fx + P)} dx$$

gdzie dx reprezentuje zmianę między poszczególnymi odcinkami do obliczenia. W algorytmie będzie ona reprezentowana poprzez krok z jakim liczona jest długość fali. W ostatnim wzorze zauważamy dużo zmiennych, których nie jesteśmy w stanie obliczyć tylko dla początkowego i końcowego x oraz zadanej długości. Dlatego algorytm musi znajdować odpowiednie współczynniki oraz obliczać dla nich długość porównując ją zadaną prawidłową (wejściową) długością, którą system musi zaprezentować.



Tabela 3.2 Opis parametrów wejściowych dla generowania sinusoid o zadanej długości

Parametr	Wartość początkowa	Opis
deltaBłędu	0.001	Próg określający dopasowanie znalezionej długości do długości pożądanej. Zwiększenie tej właściwości powoduje mniej precyzyjne rysowanie sinusoid
aInit	0.01	Parametr określający początkową amplitudę
pInit	0.1	Parametr określający początkową wartość częstotliwości. Poniżej wartości 0.1 nie ma praktycznych różnic, dlatego powstają linie proste
aMax	0.1	Maksymalna wartość amplitudy
pMax	50	Maksymalna wartość częstotliwości
aStep	0.005	Wartość kroku do znalezienia odpowiedniej amplitudy
pStep	0.1	Wartość kroku do znalezienia odpowiedniej częstotliwości sinusoidy

Algorytm wygląda następująco w pseudokodzie:

```
jeśli ( oczekiwana_długość > 0 ) {  
    dopóki ( próba <= max_ilość_prób ) {  
        dla ( p = pInit; p <= pMax; p += pStep ) {  
            ObliczWPętliDługośćSin();  
            jeśli ( ZnalezionoOczekiwanaDługość() || długość > oczekiwana_długość )  
                break;  
        }  
        jeśli ( ! (znalezionaDeltaBłędu <= deltaBłędu) ) {  
            dla ( a = aInit; a <= aMax; a += aStep ) {  
                ObliczWPętliDługośćSin();  
                jeśli ( ZnalezionoOczekiwanaDługość() || długość > oczekiwana_długość )  
                    break;  
            }  
        }  
        próba++;  
        pMax += pMax*0.5;  
        aMax *= 1.5;  
    } else  
        break;  
}
```

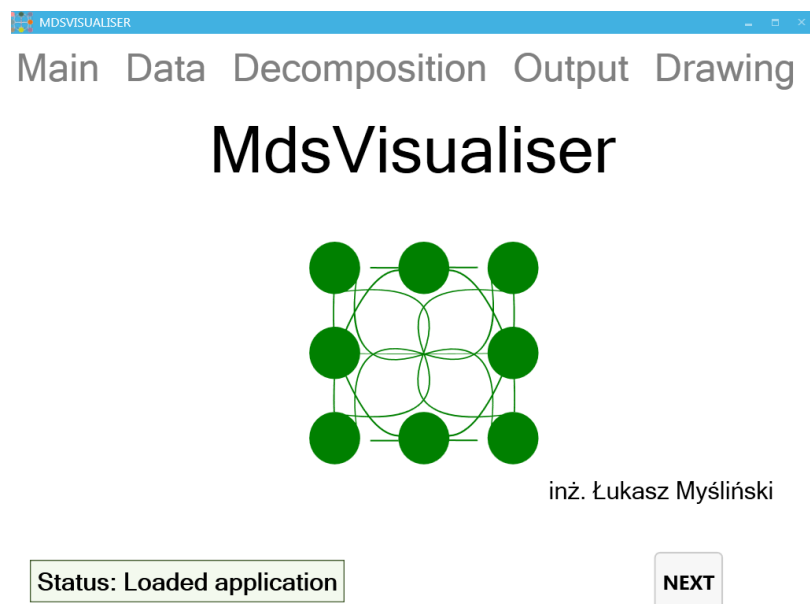


Z powyższego pseudokodu wynika, że algorytm wykonuje swoje działanie dopóki trzykrotnie nie znaleziono odpowiednich wartości w funkcji sinus. Podczas każdej próby liczona jest najpierw częstotliwość, która zostaje powiększana stopniowo i obliczana jest równocześnie długość czy, aby nie jest równa oczekiwanemu dystansowi. Jeżeli nie zostanie ona znaleziona, to algorytm wykonuje podobną operację na amplitudzie. Z kolei, jeśli w danej próbie metoda nie mogła znaleźć odpowiednich współczynników to maksymalne wartości graniczne dla amplitudy i częstotliwości zostają powiększone.

3.6 Interfejs systemu

3.6.1 Ekran główny

Aplikacja posiada ekran startowy, który zawiera logo oraz tytuł aplikacji na środku ekranu. Na samej górze znajduje się menu, które reaguje na klikanie. Z kolei na dole ekranu jest stopka z aktualną informacją lub błędem oraz w prawym dolnym brzegu znajdują się przyciski nawigacyjne.



Rysunek 3.10 Ekran główny

3.6.2 Ekran ładowania danych

Ekran ładowania danych dynamicznie reaguje na akcje użytkownika. Standardowo po uruchomieniu aplikacji wygląda następująco jak na rysunku 3.12 i umożliwia załadowanie danych w postaci macierzy odległości lub listy współrzędnych obiektów. „Split char” oznacza znak separacji danych do wyboru, domyślnie wybrana jest spacja. Jednakże można także wybrać przecinek lub



średnik. Oczywiście można też podać maksymalną ilość wymiarów do załadowania lub wygenerowania (pole tekstowe obok „Max loading dimension”). Co więcej dla danych można także załadować nagłówki (akcja „LOAD HEADER”).

Po wybraniu opcji „Generate” system zmienia dynamicznie widok, w którym użytkownik może wybrać z jakich obiektów ma wygenerować macierz dystansów.

DATA			
TYPE	DISTANCE TYPE	GENERATE OBJECT DATA	ACTIONS
<input type="radio"/> Objects <input type="radio"/> Load <input checked="" type="radio"/> Generate	<input checked="" type="radio"/> Euclidean	<input type="radio"/> Positive coords <input type="radio"/> Negative coords <input checked="" type="radio"/> Mixed coords GENERATE	Automatic <input type="checkbox"/> Max loading dimension 10

Rysunek 3.11 Ekran generowania danych

Informacje w stopce informują użytkownika o krokach, które może wykonać w aktualnym widoku. Po załadowaniu danych ekran wygląda jak na rysunku 3.12.

Main Data Decomposition Output Drawing

DATA			
TYPE	LOAD (CSV FORMAT)	ACTIONS	
<input checked="" type="radio"/> Distances <input type="radio"/> Objects <input checked="" type="radio"/> Load <input type="radio"/> Generate	Split char: LOAD SQUARE DATA LOAD HEADER	Automatic <input checked="" type="checkbox"/> Max loading dimension 4	

SOURCE								
#	SHOW	COLOR	EDIT	AU	BR	CO	DA	
Auto	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	E	0	79	53	59
Auto	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	E	79	0	67	62
Auto	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	E	53	67	0	74
Auto	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	E	59	62	74	0

Automatic is ENABLED, click next to visualization

BACK DRAW

Rysunek 3.12 Ekran po załadowaniu danych

Po załadowaniu danych aplikacja umożliwia ręczne przeprowadzenie obliczeń lub włączenie akcji „Automatic”. Po jej zaznaczeniu można od razu przejść do akcji „DRAW”, a system przeprowadzi wszelkie potrzebne obliczenia i wyświetli wizualizację.

Moduł zawiera też walidację wpisywanej wartości ładowanych lub generowanych wymiarów dla danych wejściowych, który musi być mniejszy od



stu (rysunek 3.13). Jednakże w celu komfortowego odbioru wizualizacji różnic w postaci linii zaleca się ograniczanie ilości wymiarów dla danych do piętnastu.

DATA

TYPE

☒ Distances ☐ Objects
☒ Load ☐ Generate

LOAD (CSV FORMAT)

Split char:
LOAD SQUARE DATA
LOAD HEADER

ACTIONS

Automatic: ☐
Max loading dimension:
MAKE B MATRX

SOURCE

#	EDIT	SHOW	COLOR	O1	O2	O3	O4
o1	E	<input checked="" type="checkbox"/>	<input type="color" value="#000000"/>	0	93	82	133
o2	E	<input checked="" type="checkbox"/>	<input type="color" value="#000000"/>	93	0	52	60
o3	E	<input checked="" type="checkbox"/>	<input type="color" value="#000000"/>	82	52	0	111
o4	E	<input checked="" type="checkbox"/>	<input type="color" value="#000000"/>	133	60	111	0

Max dimension is greater than 99!

BACK

Rysunek 3.13 Walidacja maksymalnego wymiaru dla ładowanych danych lub generowanych

3.6.3 Ekran dekompozycji

Na tym ekranie system wyświetla wynik EVD, jeśli został on przeprowadzony poprawnie. W przeciwnym razie stosowna informacja pojawi się w stopce oraz znikną przyciski dalszych akcji, w tym obliczeń współrzędnych (przycisk „CALCULATE COORDINATES”).

MDSVISUALISER

Main Data **Decomposition** Output Drawing

EIGEN VALUES

#	VALUES
o1	9.58
o2	1.68
o3	-0.04
o4	0.01
o5	0.01
o6	-0.01
o7	0.01
o8	0
o9	0
o10	0

EIGEN VECTORS

#	ATLANTA	CHICAGO	DENVER	HOUSTON	LOSANG.	MIAMI	NEWYORK	SANFR.	SEATTLE
Atlanta	0.23	-0.11	0.12	-0.11	-0.52	0.41	0.58	0.2	0.3
Chicago	0.12	0.26	0.15	0.26	-0.34	0.13	0.11	-0.64	-0.52
Denver	-0.16	0.02	0.03	0.59	-0.48	0.07	-0.44	0.3	0.1
Houston	0.05	-0.44	0.1	0.38	-0.1	-0.69	0.36	0.04	-0.06
LosAng.	-0.39	-0.3	-0.4	0.03	-0.15	0.07	-0.05	-0.57	0.47
Miami	0.37	-0.45	0.45	0.28	0.37	0.35	-0.22	-0.19	0.18
NewYork	0.35	0.4	-0.37	0.43	0.33	-0.04	0.31	-0.09	0.31
SanFr.	-0.46	-0.09	-0.19	0.38	0.29	0.43	0.29	0.26	-0.31
Seattle	-0.43	0.44	0.62	0.09	0.1	-0.12	0.14	-0.07	0.39
Washing.	0.32	0.26	-0.18	0.12	-0.13	-0.04	-0.28	0.13	0.16

CALCULATE COORDINATES

Calculate coordinates

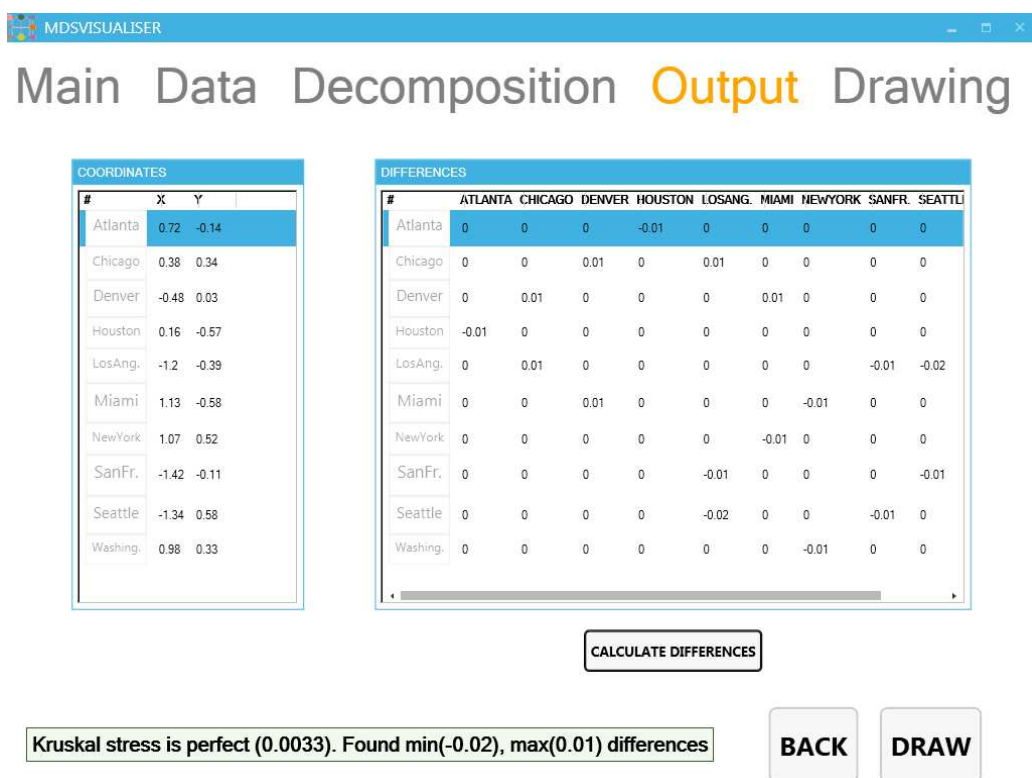
BACK **NEXT**

Rysunek 3.14 Ekran wyniku EVD



3.6.4 Ekran wynikowy

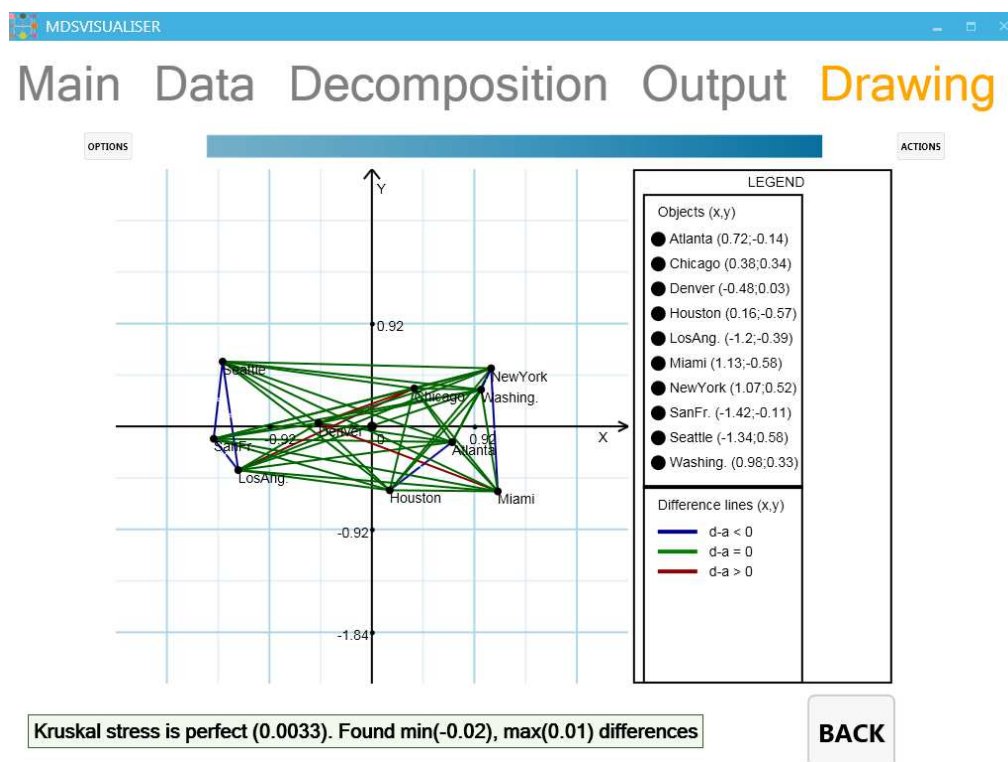
Kolejnym krokiem podczas obliczeń są wynikowe współrzędne, które system zaprezentuje na wizualizacji. Co więcej na tym etapie użytkownik może spojrzeć na poszczególne różnice dystansów między obiektami po wywołaniu akcji „CALCULATE DIFFERENCES”. Następnie może przejść do generowania grafiki poprzez wywołanie przycisku „DRAW”.



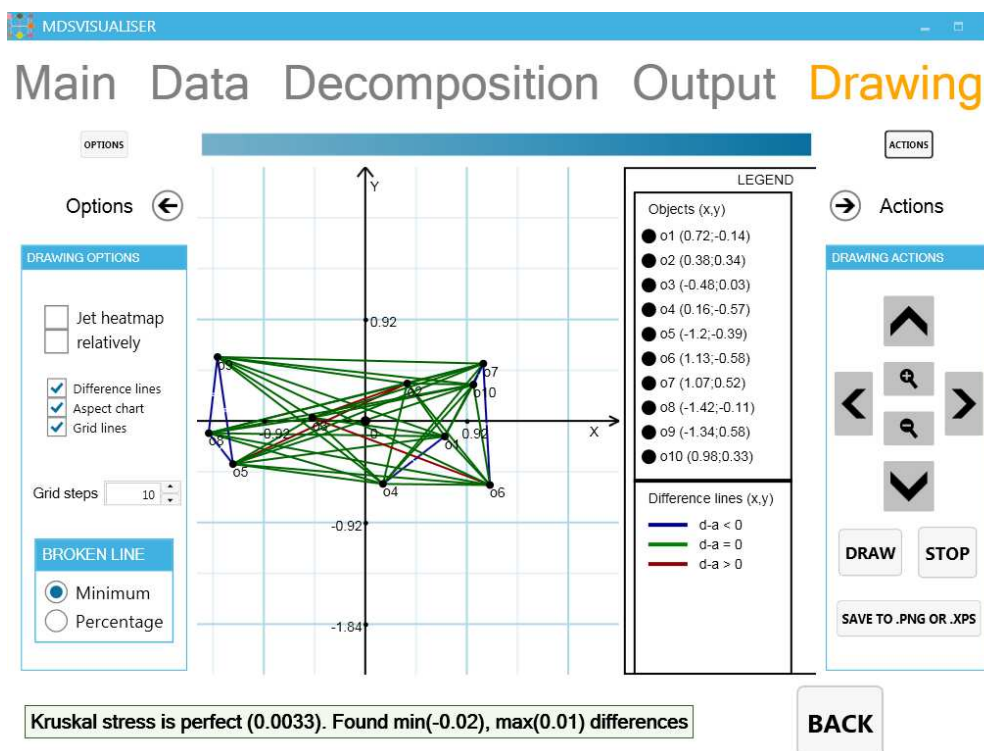
Rysunek 3.15 Ekran wynikowy systemu

3.6.5 Ekran wizualizacji

Ostatnim ekranem jest wizualizacja reprezentująca metodę MDS. Standardowy wygląda jak na rysunku 3.16. Oczywiście nie jest to koniec możliwości tego modułu. Wszystkie opcje manipulowania wizualizacją oraz wykonywania na niej akcji zostały ukryte w wysuwanych menu jak na rysunku 3.17. Opis ich działania został uwzględniony w rozdziale 3.5. Opis implementacji.



Rysunek 3.16 Standardowy ekran wizualizacji

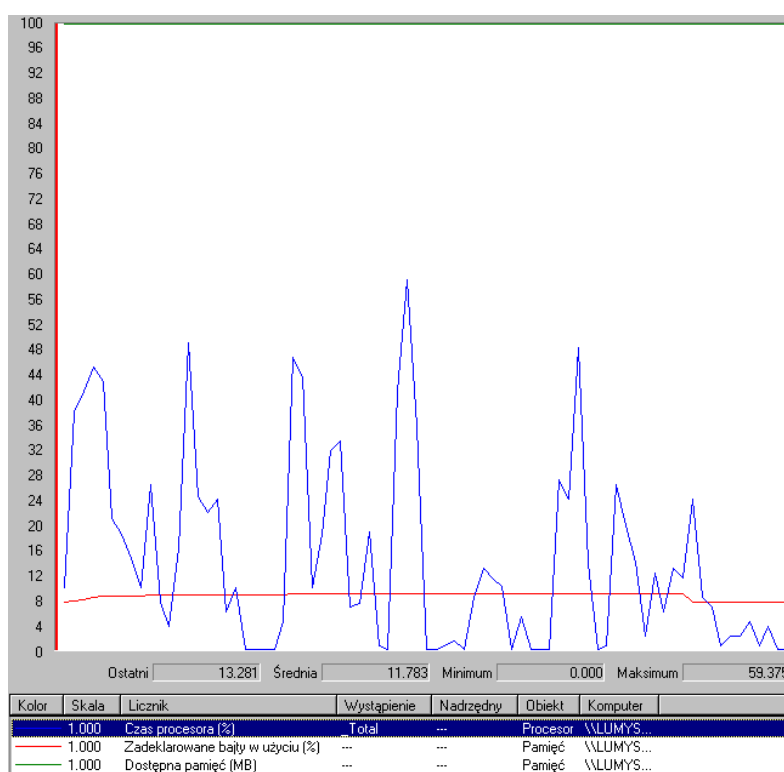


Rysunek 3.17 Standardowy ekran wizualizacji z włączonymi dodatkowymi menu

3.7 Testowanie systemu

System był testowany dla różnych parametrów generowania linii falistych. Wiązało się to z dużym zapotrzebowaniem na pamięć RAM w przypadku bardzo

niskich parametrów dla kroków podczas wyszukiwania prawidłowych współczynników dla zadanej długości krzywej. Obecnie parametry te zostały zoptymalizowane w taki sposób, że system pobiera około 50 MB pamięci podczas wizualizacji. Poniżej znajduje się wykres wydajności systemu, na którym można zaobserwować, że w momentach krytycznych takich jak uruchomienie aplikacji, ładowanie danych czy wizualizacja czas procesora liczony w procentach oscyluje do połowy swojej wartości. Oczywiście należy wziąć pod uwagę, że na systemie jest uruchomiona aplikacja wykrywająca wirusy czy inne niestandardowe usługi.



Rysunek 3.18 Wykres wydajności aplikacji

3.7.1 Testy użytkowników końcowych

System został zaprezentowany wybranej grupie losowej użytkowników. Zwracają oni uwagę na prostotę, czytelność, intuicyjność oraz przejrzystość interfejsu. Nie odwraca on uwagi użytkownika od prezentowanych funkcjonalności systemu. Jasne, stonowane kolory sprawiają, że wygląd jest estetyczny i zgodny ze współczesnymi standardami.

Z kolei najnowsze rozwiązania technologiczne pisania aplikacji sprawiają, że użytkownik ma wrażenie prawdziwej interakcji z systemem.



4. Badane zbiory

Rozdział ma na celu weryfikacji oraz przedstawienia jakości wizualizacji na podstawie przykładowych danych. Na potrzeby dalszych podrozdziałów wprowadzono oznaczenia:

- d – oznaczona odległość wejściowa (oryginalna),
- a – oznacza długość między obiektami (punktami),
- r – różnica między d i a

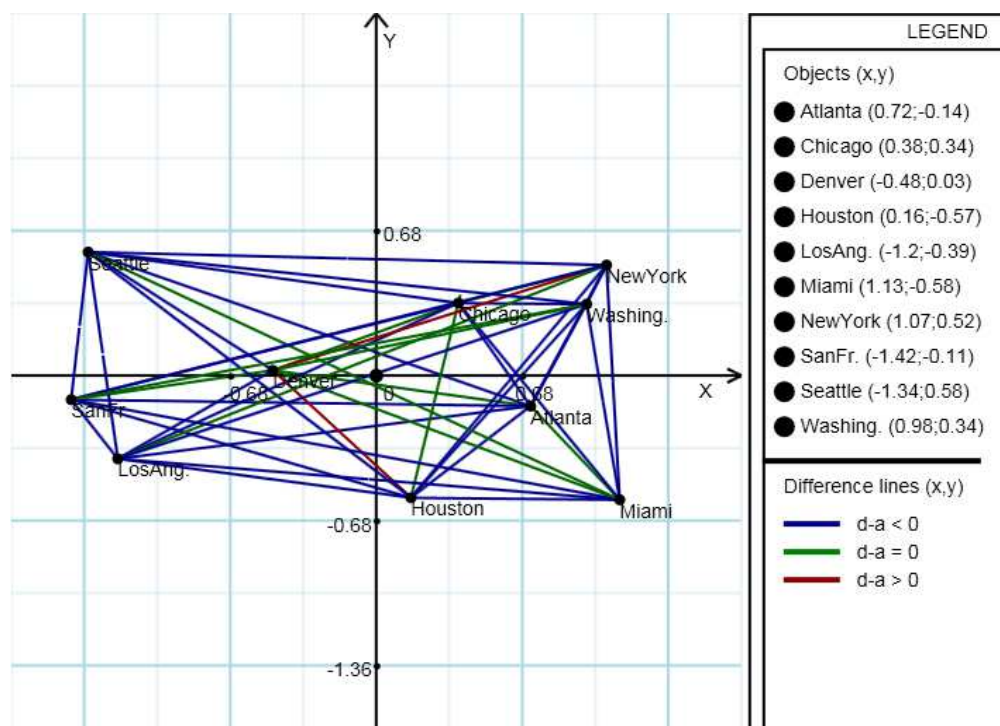
4.1 Odtwarzanie topologii na podstawie odległości

Macierz odległości lotniczych między dziesięcioma wybranymi miastami USA pochodzi z publikacji [KRUS 1978]. Trudno sobie wyobrazić topologię geograficzną lokalizacji tych miast na podstawie poniższych danych.

Tabela 4.1 Odległości lotnicze między dziesięcioma wybranymi miastami w USA

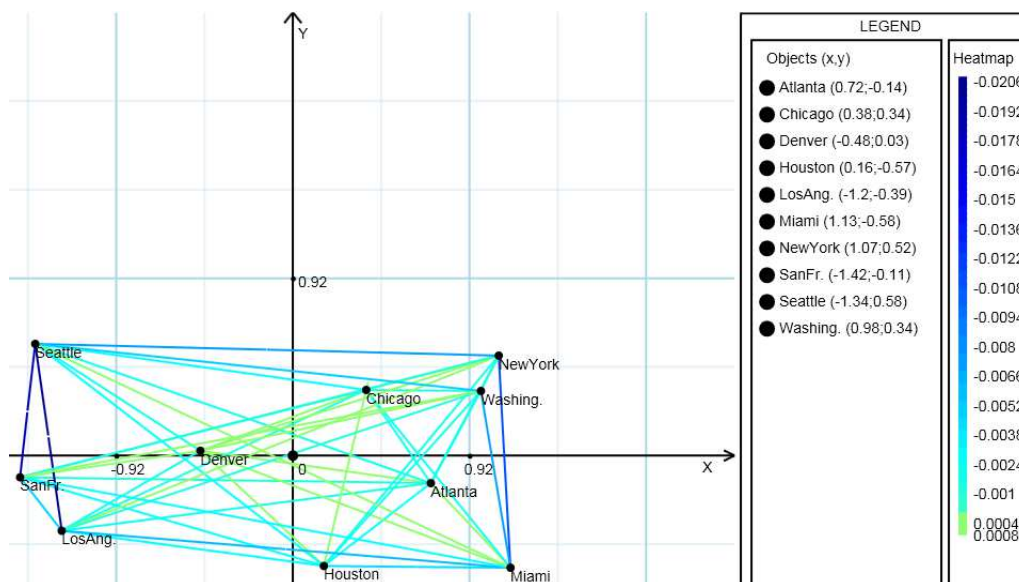
	Atlanta	Chicago	Denver	Houston	Los Angeles	Miami	New York	San Francisco	Seattle	Washington
Atlanta	0	587	1212	701	1936	604	748	2139	2182	543
Chicago	587	0	920	940	1745	1188	713	1858	1737	597
Denver	1212	920	0	879	831	1726	1631	949	1021	1494
Houston	701	940	879	0	1374	968	1420	1645	1891	1220
Los Angeles	1936	1745	831	1374	0	2339	2451	347	959	2300
Miami	604	1188	1726	968	2339	0	1092	2594	2734	923
New York	748	713	1631	1420	2451	1092	0	2571	2408	205
San Francisco	2139	1858	949	1645	347	2594	2571	0	678	2442
Seattle	2182	1737	1021	1891	959	2734	2408	678	0	2329
Washington	543	597	1494	1220	2300	923	205	2442	2329	0

Metoda MDS na bazie tej macierzy wartości potrafi odwzorować współrzędne z jakich powstały na rysunku 4.1 poniżej.



Rysunek 4.1 Wizualizacja przykładu odległości lotniczych miast USA

Błąd, który powstał po redukcji jest prawie równy zeru (0.0033) o czym świadczy duża ilość zielonych linii. Wynik tej reprezentacji można poprawić poprzez zastosowanie relatywnej mapy kolorów. W ten sposób możemy jasno stwierdzić, gdzie są różnice wynikłe z zastosowanego skalowania wymiarów.



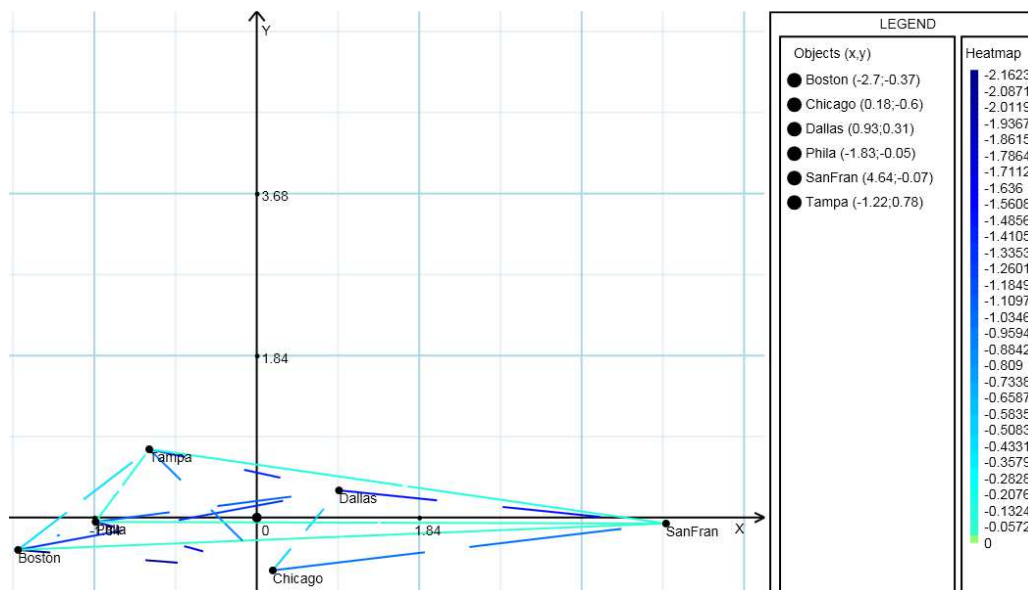
Rysunek 4.2 Wizualizacja danych odległości lotniczych z użyciem relatywnej mapy kolorów

Kolejnym dość ciekawym przykładem jest zbiór dystansów pochodzący z pracy [DUBI 2001]. Autorzy tej publikacji postanowili każdą odległość między miastami podnieść do kwadratu po czym podzielić przez dziesięć. Na przykład

z Bostonu do Chicago jest 856 mil ($856^2/10 = 732736/10 = 7327.36$). Z tego powodu wizualizacja poniżej nie odzwierciedla w pełni odległości na podstawie, których macierz w tabeli 4.2 powstała. To przekształcenie bardzo zaburza metodę klasycznego skalowania przez co wartość błędu jest duża (wynosi 0.2862). Jednak idealnie nadaje się do zaprezentowania ujemnych różnic w dystansach między poszczególnymi obiektami.

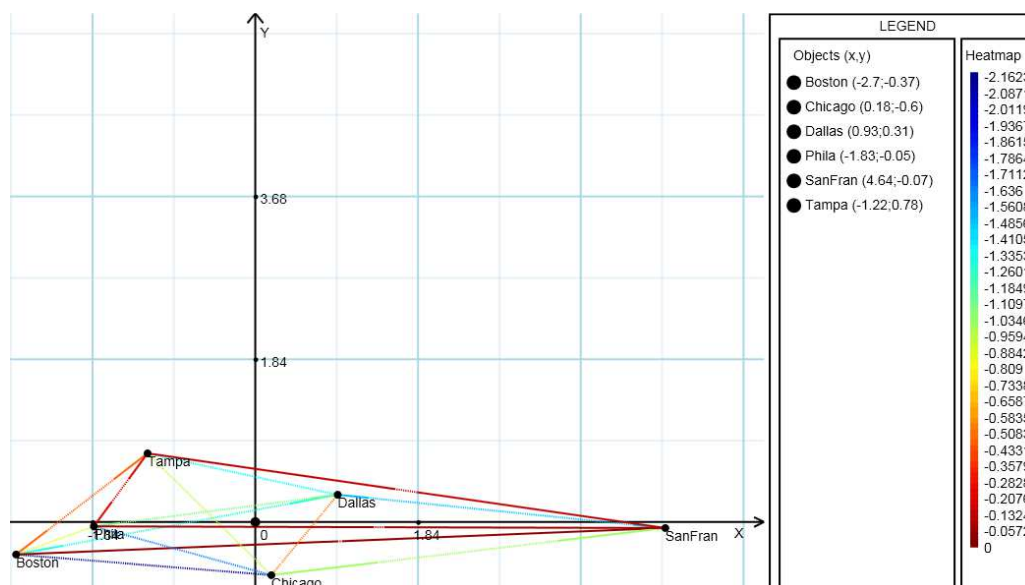
Tabela 4.2 Macierz odległości wybranych miast USA

	Boston	Chicago	Dallas	Phila	SanFran	Tampa
Boston	0	7327	24056	718	73332	13971
Chicago	7327	0	6368	4462	34707	10020
Dallas	24056	6368	0	16900	22290	8372
Phila	718	4462	16900	0	64009	8667
SanFran	73332	34707	22290	64009	0	57936
Tampa	13971	10020	8372	8667	57936	0



Rysunek 4.3 Wizualizacja rekonstrukcji mapy USA na bazie różnych miast

W tym momencie dobrze byłoby zobaczyć wizualizację z włączonym algorytmem generowania linii przerywanych na bazie absolutnej wartości z minimalnej różnicy (w tym wypadku wynosi ona -0.013). Z uwagi na bardzo niską wartość, system wyrysował bardzo krótkie przerwy (rysunek 4.4), które są widocznie prawie jak kropki. Jednak po wyłączeniu relatywnej mapy kolorów efekt poprawił się przez zastosowanie większej palety kolorów i można jasno stwierdzić, że długie linie są prawie równe zero. Wszystkie ujemne różnice wynikają prawdopodobnie z powodu nietypowej macierzy wejściowej.



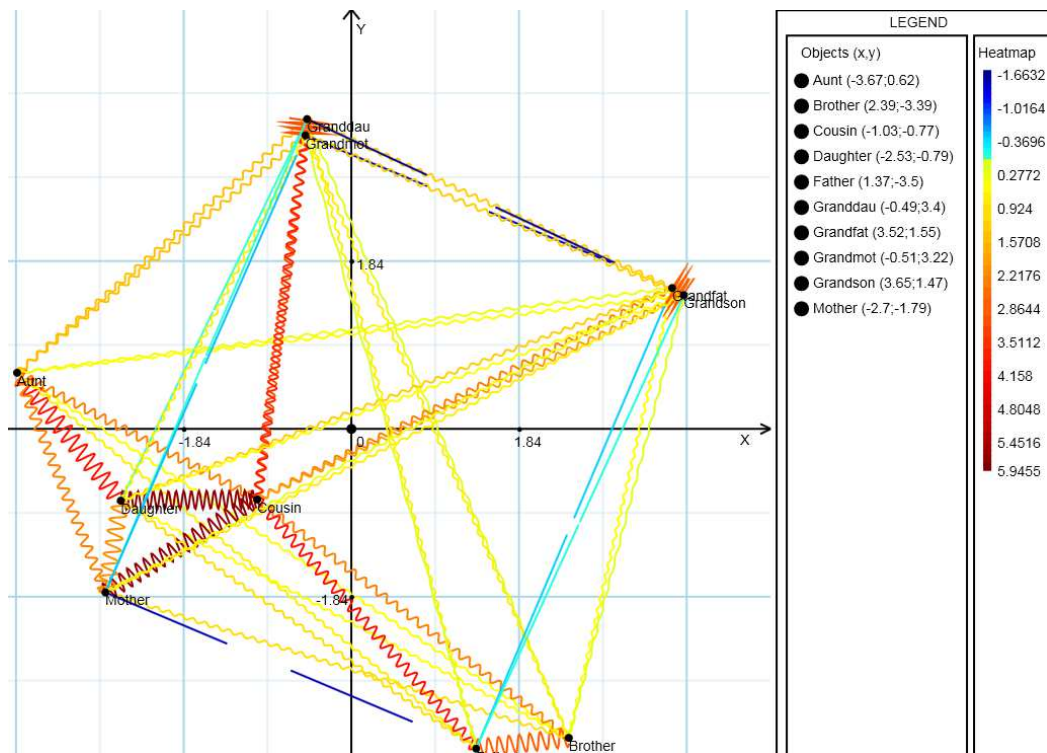
Rysunek 4.4 Wizualizacja rekonstrukcji mapy USA z użyciem algorytmu „minimum”

4.2 Odtwarzanie podobieństwa między obiektami

Rosenberg i Kim przeprowadzili eksperyment pośród studentów, którzy określali stopień podobieństwa między pokrewieństwem. Dane są wartościami procentowymi i pochodzą ze źródła [DATA]. Jest to kolejny przykład na to, że pomimo dużego błędu metody MDS (0.42) wizualizacja dość dobrze prezentuje różnice, dzięki rysowaniu linii.

Z rysunku 4.5 można łatwo wywnioskować, że różnice między obiektami Granddau i Grandmat są takie same jak między Grandfat i Grandson. Podobnie sprawa wygląda w przypadku obiektów Daughter i Cousin oraz Mother i Cousin.

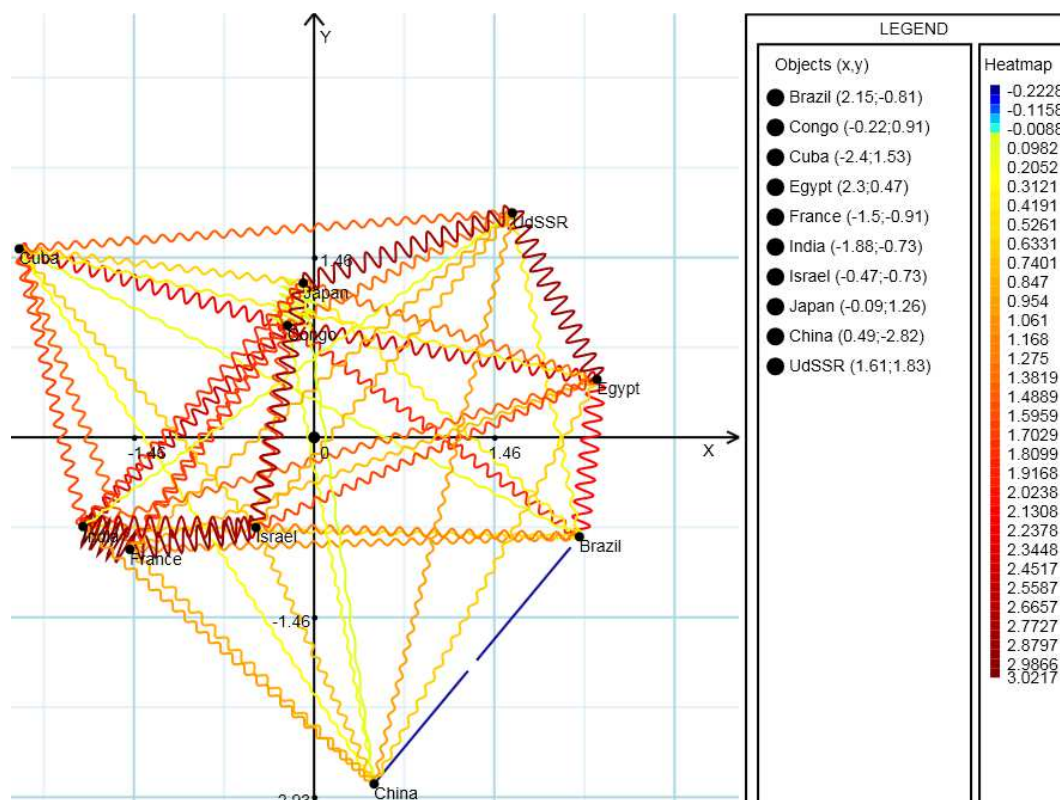
Oryginalne dane mają piętnaście wymiarów, jednak zostało załadowanych pierwszych dziesięć wymiarów w celu komfortowego odbioru wizualizacji.



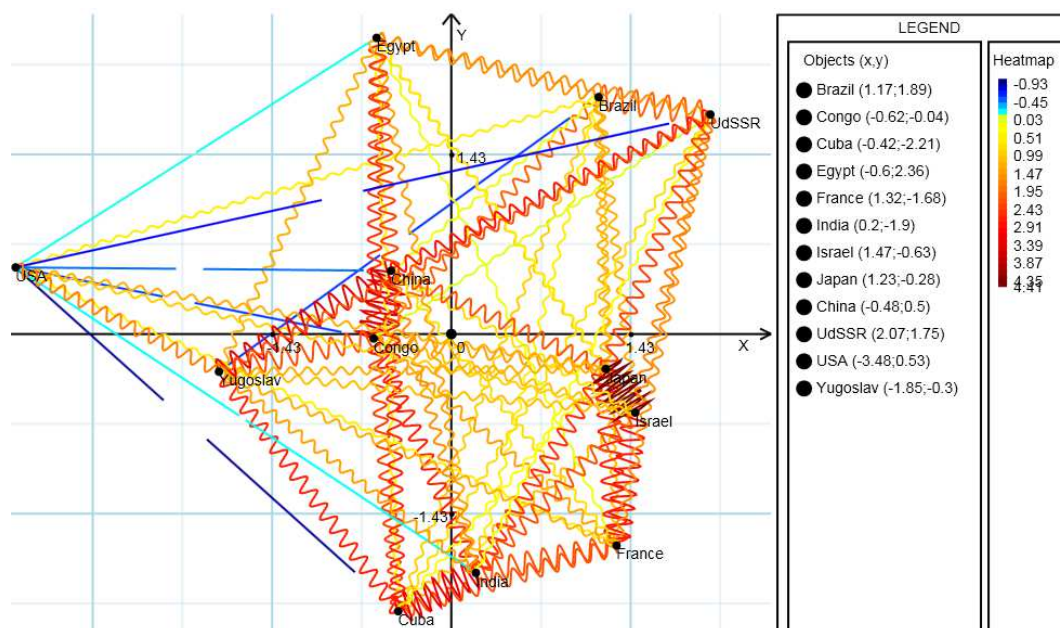
Rysunek 4.5 Wizualizacja podobieństwa między pokrewieństwem

Następny przykład danych tego typu to podobieństwo między państwami. Wish poprosił osiemnastu studentów o ocenę podobieństw w skali jeden do dziewięć (gdzie jeden oznacza bardzo różne, a dziewięć bardzo podobne). Dane te pochodzą również z tego samego źródła [DATA]. Wynik znalezionego rozwiązania jest najślabszy w tym rozdziale.

Rysunek 4.6 prezentuje dziesięć z dwunastu wymiarów (błąd 0.53). Dzięki wizualizacji linii i ich kolorowania można zauważyć, że największa różnica w odwzorowaniu odległości jest między *India*, a *France* ($d=3.439, a=0.424, r=3.016$). Następna duża różnica jest pomiędzy *Japan*, a *Israel* ($d=4.83, a=2.026, r=2.804$). Jednakże po załadowaniu pełnych danych wynik metody MDS zmienia współrzędne niektórych obiektów tak jak to widać na rysunku 4.7. Błąd z 0.53 zwiększył się do 0.60. Dystanse wyjściowe też uległy zmianie, w ten sposób różnica między *Japan*, a *Israel* zwiększyła się ($d=4.83, a=0.42, r=4.41$). Zmniejszyła się za to między *India*, a *France* ($d=3.439, a=1.1143, r=2.297$).



Rysunek 4.6 Wizualizacja podobieństwa między dziesięcioma krajami (częściowe dane)



Rysunek 4.7 Wizualizacja podobieństwa między dwunastoma krajami (pełne dane)

5. Podsumowanie

Kwintesencją prac nad stworzeniem systemu do wizualizacji metody MDS jest rozdział poświęcony badanym zbiorom. Oddaje to częściowo obraz i otrzymany efekt, który stanowi tylko małą próbkę możliwości analitycznych i syntetycznych jakie może zaoferować ten system w badaniach psychologicznych oraz z zakresu nauk społecznych, humanistycznych czy przyrodniczych.

Skalowanie wielowymiarowe nie tylko stosujemy po to, aby po prostu zmniejszyć ilość wymiarów czy wyłonić pewną klasyfikację. Jednak, jak zostało to podkreślone w podsumowaniu publikacji [GRAM 2009] najważniejsze, że w wyniku tej metody ukazuje się często więcej informacji niż przed jej zastosowaniem. Odkrywane są wtedy relacje łączące obiekty, które ten system stara się dodatkowo podkreślić poprzez odległości nieeuklidesowe. Szczególnie jest to przydatne w przypadku danych, z których nasuwają się niejednoznaczne decyzje. Konflikty takie zdarzają się niejednokrotnie między osobami, które są ekspertami z różnych dziedzin naukowych. Dlatego system ten może być cennym narzędziem komunikacji dla nich [BIELA 1992].

Pomimo tego, metoda MDS nie może być rozwiązaniem dla wszystkich problemów metodologicznych. Co więcej zmniejszaniu wymiarów powinno się poddawać tylko te dane, które są warte poniesionych kosztów. Metoda ta jest bardziej nastawiona na wykrywanie pewnych struktur w badanych procesach, zachowaniach, faktach czy związkach [BIELA 1992]. Szczególnie wersja klasyczna idealnie nadaje się tylko dla danych, które są odległościami lub wartością podobieństw między obiektami. Im więcej pełniejszych danych system jest w stanie przetworzyć, tym mniejszy błąd i mniej falistych oraz przerywanych linii na wizualizacji. Przyjmując oczywiście, że dane te powstały w modelu euklidesowym.

Rozdział ten został poświęcony weryfikacji na ile system dobrze zrealizował swoje zadanie oraz do jakich praktycznych celów może zostać użyty.

5.1 Weryfikacja użytkowa

System do wizualizacji metody MDS jest wbrew pozorom złożoną aplikacją w miarę zwiększania jej parametrów wejściowych oraz możliwości, które oferuje



najnowsza technologia. Aplikacja rozwiązuje problem, który był celem pracy zachowując jego prostotę, czytelność i przejrzystość interfejsu.

Podczas wykonywania badań i testów okazało się, że metoda SVD pierwotnie użyta do dekompozycji macierzy w metodzie MDS nadaje się tylko do szczególnych przypadków, w których dane wejściowe powstają z konfiguracji obiektów położonych w przestrzeni nieujemnej. Dlatego też po konsultacji nastąpiła zmiana na rozkład EVD, który jest pozbawiony tego typu ograniczeń.

Przy jego tworzeniu opracowywano nowe metody bazując na istniejących rozwiązaniach czy przemyśleniach takie jak: kontrolka do wyświetlania dowolnej macierzy danych, nawigacja w stylu kreator (ang. wizard) [WIZARD], minimalistyczne menu, relatywna mapa kolorów oraz inne rozszerzenia standardowych klas.

W obecnej wersji systemu nie ma możliwości edytowania danego wiersza danych z poziomu widoku czy też ustawiania widzialności i koloru dla wybranego obiektu. Dodatkowo nie ma walidacji ładowanych macierzy symetrycznych pod kątem integralności danych. Z powodu ograniczeń czasowych ta część została pominięta.

Aplikacja na tle omówionych we wstępie konkurentów oraz innych programów, które do tej pory powstały, wyróżnia się innowacyjnym podejściem do tematyki reprezentowania błędu powstałego po zastosowaniu metody MDS.

Oferuje możliwość interakcji poprzez zaznaczanie obiektów oraz linii czy też dzięki manipulacji wizualizacją poprzez przesuwanie, zmniejszanie lub powiększanie. Wprowadza nowe standardy oraz możliwości poprzez generowanie poszczególnych typów linii między obiektami w celu wizualizacji prawidłowych długości z macierzy wejściowej. Co więcej system zwiększa poziom odbioru i precyzjności, dzięki zastosowaniu relatywnej mapy kolorów reprezentująca składowe błędu. Mówiąc dokładniej poszczególnych różnic dystansów między długością zadaną w macierzy wejściowej, a wyjściową obliczoną z konfiguracji współrzędnych w wyniku metody MDS.

Podsumowując, system spełnia swoje główne zadanie przy zachowaniu minimalizmu i maksymalnej czytelności oraz ukazuje dalsze możliwości rozwoju tego systemu.



5.2 Rozbudowa systemu

Opisywany system z pewnością można rozwinąć do kolejnych wersji. Pierwszym krokiem byłaby implementacja już gotowego widoku ładowania danych na którym pozostało do skończenia:

- edycja danych w wierszach,
- ustawianie koloru dla wybranego obiektu,
- ustawienie widzialności dla wybranego obiektu.

Dodatkowo w module wizualizacji można dodać więcej interakcji takich jak:

- rotacja całej wizualizacji,
- ustawianie widzialności obiektów,
- przeciąganie obiektów w celu minimalizacji powstałego błędu, poprzez:
 - prostowanie lin falistych,
 - zmniejszanie przerw w liniach przerywanych.

W tym miejscu warto wspomnieć o pomysłe stworzenia gry edukacyjnej. Przy wykorzystaniu najnowszych technologii w telefonach, jaką jest na przykład dotykowy ekran, można zachęcić i zainteresować zwykłych użytkowników do skorzystania z rozwiązań zawartych w pracy.

Co więcej, można pokusić się o wystawienie interfejsu programowania aplikacji (ang. Application Programming Interface, API) w celu opublikowania i wykorzystania przez osoby trzecie. W ten sposób można dosyć łatwo powiązać go z systemem sieci społecznościowych, co umożliwiłoby zaprezentowanie bliskich nam na przykład dziesięciu osób. Z kolei zaś wyrysowane linie uprościłyby podjęcie decyzji w przypadkach, które wydają się konfliktowe.

Jest oczekiwane, że metoda MDS będzie generowała dalsze zainteresowanie na polu rozwoju coraz bardziej elastycznych i niezawodnych algorytmów. Z pewnością pozostanie potężną i użyteczną metodologią w naukach społecznych i poznawczych.



6. Bibliografia

- [.NET] http://pl.wikipedia.org/wiki/.NET_Framework
- [ASSEMBLA] <https://www.assembla.com/home>
- [BELK 2002] Mikhail Belkin, Partha Niyogi, „Laplacian eigenmaps for dimensionality reduction and data representation”, 2002
- [BIELA 1992] Adam Biela, „Skalowanie wielowymiarowe jako metoda badań naukowych”, 1992, s. 7, s. 106
- [BORG 2005] Borg, I. and Groenen, P.J.F. (2005). Modern multidimensional scaling. 2nd edition. New York: Springer., s. 41
- [BOURNE] Murray Bourne, <http://www.intmath.com/applications-integration/11-arc-length-curve.php>
- [BUJA 2001] Andreas Buja, Deborah F. Swayne, Michael L. Littman, Nathaniel Dean i Heike Hofmann, XGvis: Interactive Data Visualization with Multidimensional Scaling, 2001
- [C++] <http://pl.wikipedia.org/wiki/C%2B%2B>
- [CSHARP] http://pl.wikipedia.org/wiki/C_Sharp
- [DATA] http://cda.psych.uiuc.edu/mds_509_2013/borg_groenen/data_mmds/HTML/
- [DELP] <http://pl.wikipedia.org/wiki/Delphi>
- [DESKTOP] <http://pl.wikipedia.org/wiki/Oprogramowanie>
- [DIRECTX] <http://pl.wikipedia.org/wiki/DirectX>
- [DOTN] <http://www.dotnumerics.com/>
- [DUBI 2001] Dave Dubin, Classical Metric Multidimensional Scaling, 2001, s.3-4
- [FORTRAN] <http://en.wikipedia.org/wiki/Fortran>
- [GASH 2009] Mike Gashler, Dan Ventura, Tony Martinez, „Iterative Non-linear Dimensionality Reduction by Manifold Sculpting”, 2009, s. 1
- [GDI] [http://msdn.microsoft.com/en-us/library/windows/desktop/dd145203\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd145203(v=vs.85).aspx)
- [GGOBI] <http://www.ggobi.org/>
- [GOWE 1966] Gower J C., Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53:325-38, 1966



[GRAM 2009] Jarosław Gramacki, Artur Gramacki, Redukcja wymiarowości oraz wizualizacja danych wielowymiarowych z wykorzystaniem projektu R, 2009, s. 209, s. 227

[HEAD 2010] Dr. Ronald B. Heady, Dr. Jennifer L. Lucas, Agnes Scott College, „Mds Analysis Using Permap 11.8”, 2010

[ICOFX] <http://icofx.ro/>

[IDC 2011] IDC's Digital Universe Study sponsored by EMC Corporation, <http://www.emc.com/leadership/programs/digital-universe.htm>, 2011

[IDC 2013] IDC's Digital Universe Study sponsored by EMC Corporation, <http://www.emc.com/leadership/digital-universe/iview/executive-summary-a-universe-of.htm>, 2013

[INK] <http://www.inkscape.org/en/download/>

[JAMR 2001] Dariusz Jamróz, Patrzenie w przestrzeni n-wymiarowej, Computer Science. Vol. 3. 2001, s. 95-96

[JAVA] <http://pl.wikipedia.org/wiki/Java>

[KEIM 2002] Keim D.A., Information Visualization and Visual Data Mining, IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 7, NO. 1, 2002, s. 100

[KRUS 1964] Kruskal, J. B., Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 29:1-27, 1964

[KRUS 1978] Kruskal, J. B. and Wish, M., Multidimensional Scaling, Sage University Paper Series on Quantitative Applications in the Social Sciences, 07-011, Beverly Hills and London: Sage Publications, 1978

[LAPACK] <http://en.wikipedia.org/wiki/LAPACK>

[LEEU 2000] de Leeuw. Jan, Multidimensional Scaling, 2000, s. 1-3

[METRO] <http://mahapps.com/MahApps.Metro/>

[MMDS 2007] Modern Multidimensional Scaling Software, http://people.few.eur.nl/groenen/mmds/mds_software/index.html, 2007

[MOQUPS] <https://moqups.com/>

[MVC] <http://pl.wikipedia.org/wiki/Model-View-Controller>

[NLOG] <http://nlog-project.org/>

[PASC] http://pl.wikipedia.org/wiki/Object_Pascal



[RIYA 2013] Dr. Riyazuddin Qureshi, Yahya Mohammad AlManna & Dr Anwar Pasha Deshmukh, Big Data: Growing pressure on global storage by data created on Social Networking Sites, International Journal of Computer Science and Management Research, 2013, s. 1876

[SAUL 2000] Lawrence K. Saul, Sam T. Roweis, „An Introduction to Locally Linear Embedding” 2000

[SHEP 1962] Shepard, R. N., The analysis of proximities: multidimensional scaling with an unknown distance function. Psychometrika, 27:125-140; 219-246, 1962

[SVN] <http://pl.wikipedia.org/wiki/Subversion>

[SYNC] <http://www.syncfusion.com/downloads/metrostudio>

[TAKA 2009] Yoshio Takane, Sunho Jung, and Yuriko Oshima-Takane, Multidimensional Scaling, s.9-15, 2009

[TATE 2010] Karl Tate, Adam „Hadhazy Zettabytes Now Needed to Describe Global Data Overload”, <http://www.technewsdaily.com/425-zettabytes-now-needed-to-describe-global-data-overload.html>, 2010

[TENE2000] Joshua B. Tenenbaum, Vin de Silva, John C. Langford, „A Global Geometric Framework

[TORG 1952] Torgerson, W.S., Multidimensional scaling: I. Theory and method. Psychometrika 17, 401-419, 1952

[TORG 1958] Torgerson, W.S., Theory and Methods of Scaling, Wiley, 1958

[TSVN] <http://tortoisesvn.net/>

[VISIO] http://pl.wikipedia.org/wiki/Microsoft_Visio

[VS] http://pl.wikipedia.org/wiki/Microsoft_Visual_Studio

[VUYK 2007] Denis Vuyka, „WPF DIAGRAMMING. SAVING YOU CANVAS TO IMAGE, XPS DOCUMENT OR RAW XAML.”, Blog <http://denisvuyka.wordpress.com/2007/12/03/wpf-diagramming-saving-you-canvas-to-image-xps-document-or-raw-xaml/>, 2007

[WARE 2006] Ware C., 2006, Information Visualization, perception for design, second edition, Elsevier, s. 2

[WEBS 2010] Doug Webster, IP Traffic to Quadruple by 2015, Cisco Blog <http://blogs.cisco.com/sp/ip-traffic-to-quadruple-by-2015/>, 2011



[WEIN 2004] K. Q. Weinberger, F. Sha, L. K. Saul, „Learning a kernel matrix for nonlinear dimensionality reduction”, 2004

[WINXP] http://pl.wikipedia.org/wiki/Windows_XP

[WIZARD] [http://en.wikipedia.org/wiki/Wizard_\(software\)](http://en.wikipedia.org/wiki/Wizard_(software))

[WOLF] <http://www.wolframalpha.com/>

[WPF] http://en.wikipedia.org/wiki/Windows_Presentation_Foundation

[WSKA] <http://pl.wikibooks.org/wiki/C/Wska%C5%BAniki>

[XML] <http://pl.wikipedia.org/wiki/XML>

[XU 2009] Jack Xu, „Practical WPF Charts and Graphics”, 2009, s.147-155

[YOUN 1984] Forrest W. Young, „Scaling”, 1984, s. 55

[YOUN 1987] Forrest W. Young, Robert M. Hamer, Multidimensional Scaling: History, Theory, and Applications for Nonlinear Dimensionality”, SCIENCE 290 (5500) VOL 290, 2000, s. 2319, 1987, s.1



7. Dodatki

7.1 Spis ilustracji

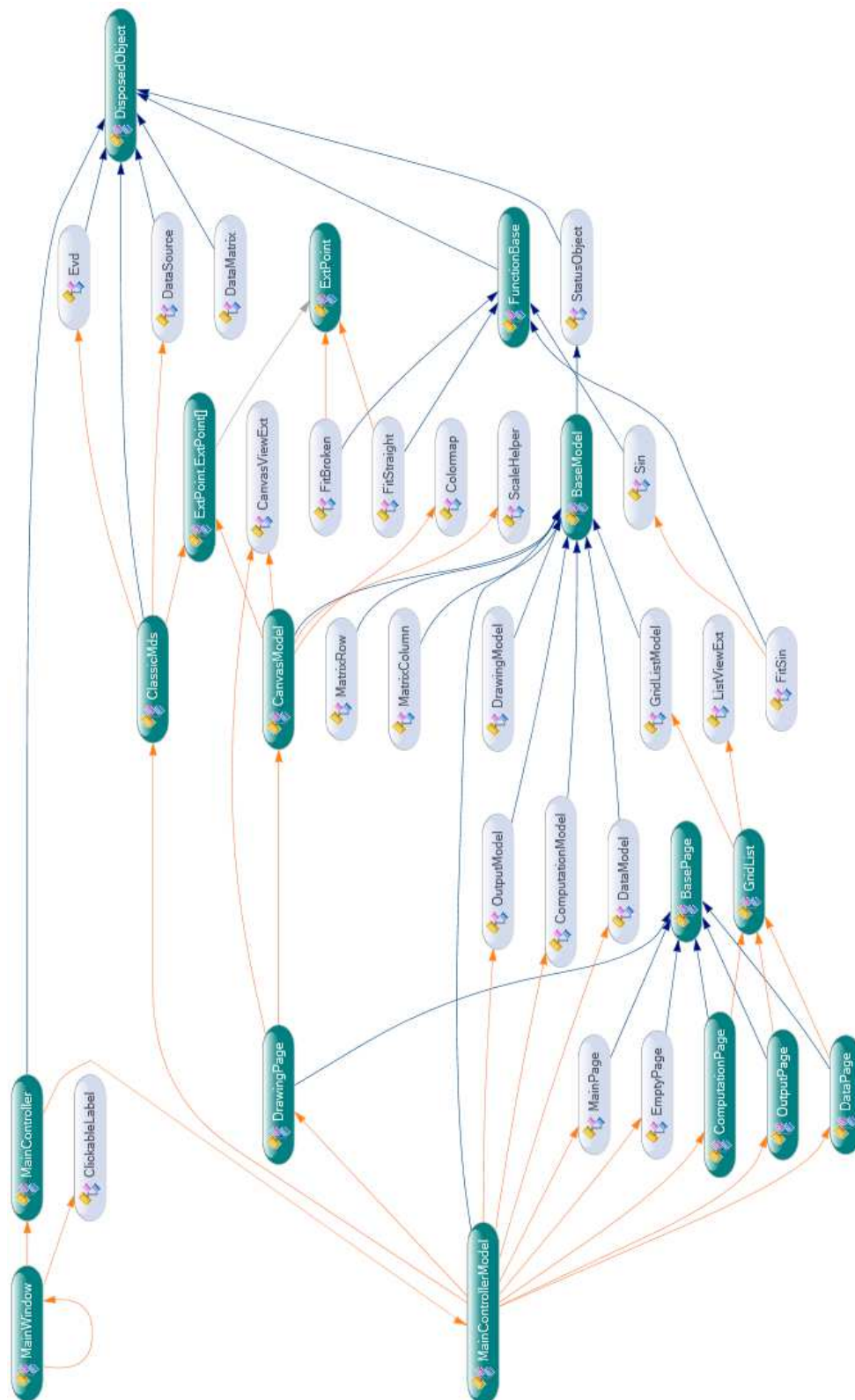
Rysunek 1.1 Część infografiki ukazującej skalę jednostki zettabajt [WEBS 2010]	5
Rysunek 1.2 Porównanie wygenerowanych ilości danych w raporcie „The Digital Universe” [IDC 2011]	5
Rysunek 1.3 Stosunek ceny wszystkich inwestycji (żółta linia) do analizy, utrzymywania i zarządzania jednym gigabajtem danych (niebieska linia) [IDC 2012]	6
Rysunek 1.4 Główny widok dodatku ggvis	9
Rysunek 1.5 Widok w module ggvis po uruchomieniu metody MDS	10
Rysunek 1.6 Zrzut ekranu z programu PerMap	12
Rysunek 2.1 Reprezentacja graficzna szukanego wyniku metody MDS	21
Rysunek 3.1 Różnice dystansów między poszczególnymi obiektami w wyniku MDS	24
Rysunek 3.2 Ogólna wartość błędu określająca jakość rozwiązania MDS	24
Rysunek 3.3 Widok menu aplikacji	32
Rysunek 3.4 Przykład macierzy symetrycznej	33
Rysunek 3.5 Przykład współrzędnych do załadowania	33
Rysunek 3.6 Zaznaczenie linii falistej	35
Rysunek 3.7 Przykładowa wizualizacja z włączonym algorytmem „minimum”	38
Rysunek 3.8 Przykładowa wizualizacja z włączonym algorytmem „percentage”	39
Rysunek 3.9 Częściowa długość fali [BOURNE]	40
Rysunek 3.10 Ekran główny	42
Rysunek 3.11 Ekran generowania danych	43
Rysunek 3.12 Ekran po załadowaniu danych	43
Rysunek 3.13 Walidacja maksymalnego wymiaru dla ładowanych danych lub generowanych	44
Rysunek 3.14 Ekran wyniku EVD	44
Rysunek 3.15 Ekran wynikowy systemu	45
Rysunek 3.16 Standardowy ekran wizualizacji	46
Rysunek 3.17 Standardowy ekran wizualizacji z włączonymi dodatkowymi menu	46
Rysunek 3.18 Wykres wydajności aplikacji	47
Rysunek 4.1 Wizualizacja przykładu odległości lotniczych miast USA	49
Rysunek 4.2 Wizualizacja danych odległości lotniczych z użyciem relatywnej mapy kolorów	49
Rysunek 4.3 Wizualizacja rekonstrukcji mapy USA na bazie różnych miast	50
Rysunek 4.4 Wizualizacja rekonstrukcji mapy USA z użyciem algorytmu „minimum”	51
Rysunek 4.5 Wizualizacja podobieństwa między pokrewieństwem	52
Rysunek 4.6 Wizualizacja podobieństwa między dziesięcioma krajami (częściowe dane)	53
Rysunek 4.7 Wizualizacja podobieństwa między dwunastoma krajami (pełne dane)	53
Rysunek 7.1 Model obiektowy głównych obiektów	62
Rysunek 7.2 Model obiektowy modeli danych dla stron	63
Rysunek 7.3 Model obiektowy stron	64
Rysunek 7.4 Model obiektowy funkcji różnych typów linii	65
Rysunek 7.5 Model obiektowy głównego okna	65
Rysunek 7.6 Przypadki użycia podczas obsługi danych	66
Rysunek 7.7 Przypadki użycia podczas obsługi wizualizacji	68
Rysunek 7.8 Infografika ukazująca skalę jednostki zettabajt [TATE 2010]	71
Rysunek 7.9 Pełna wersja infografiki ukazującej skalę jednostki [WEBS 2010]	72

7.2 Spis tabel

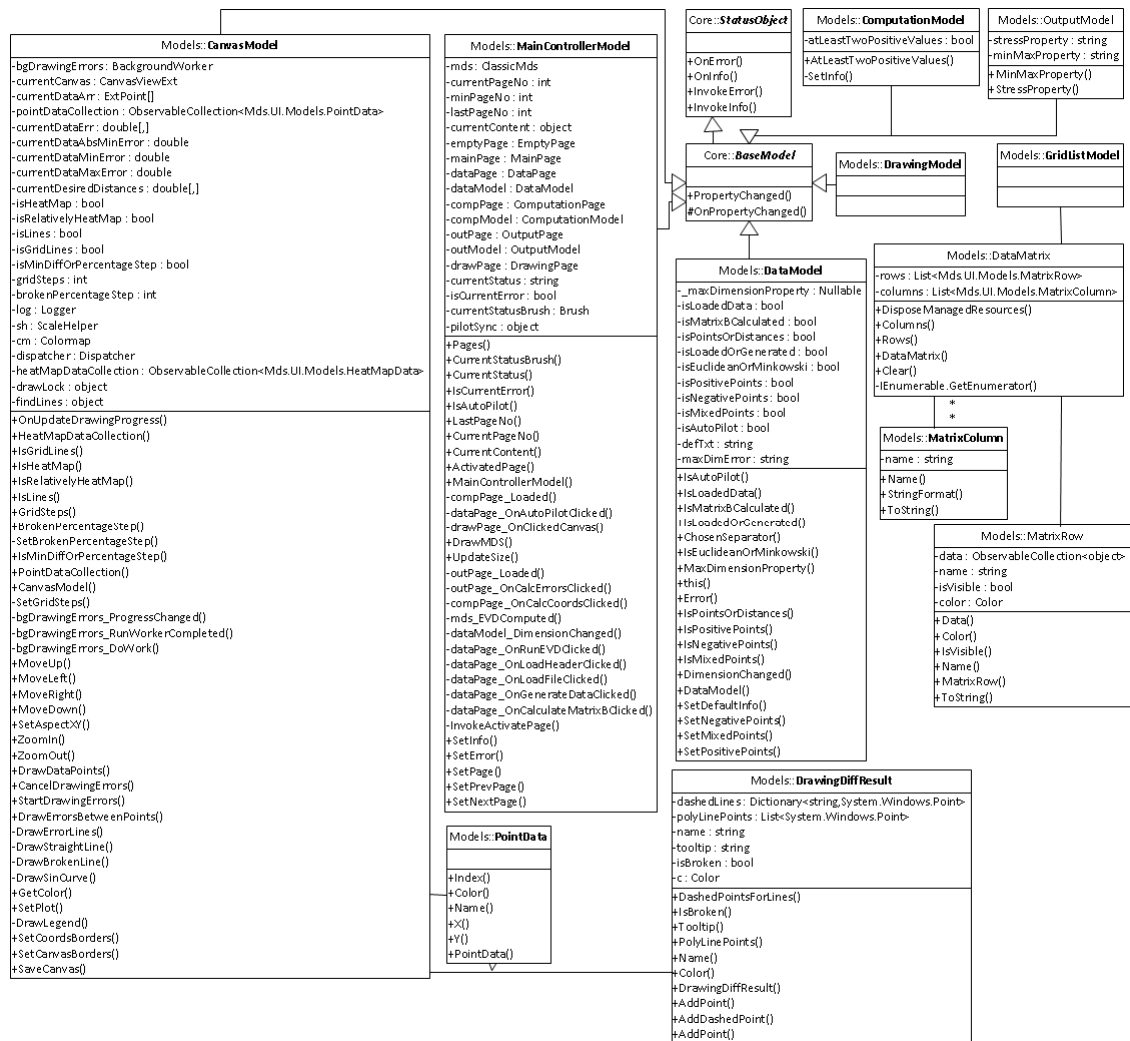
Tabela 1.1 Spis oprogramowania oferującego skalowanie wielowymiarowe [MMDS 2007]	8
Tabela 2.1 Przykładowa macierz odległości	20
Tabela 2.2 Wartość funkcji kosztu oraz odpowiadające mu oznaczenie	23
Tabela 3.1 Porównanie mapy kolorów „Jet”	36
Tabela 3.2 Opis parametrów wejściowych dla generowania sinusoid o zadanej długości	41
Tabela 4.1 Odległości lotnicze między dziesięcioma wybranymi miastami w USA	48
Tabela 4.2 Macierz odległości wybranych miast USA	50
Tabela 7.1 Opis tabelaryczny dla przypadków użycia podczas obsługi danych	66
Tabela 7.2 Opis tabelaryczny dla przypadków użycia podczas obsługi wizualizacji	68
Tabela 7.3 Pseudokod algorytmu „minimum”	70

7.3 Diagramy klas

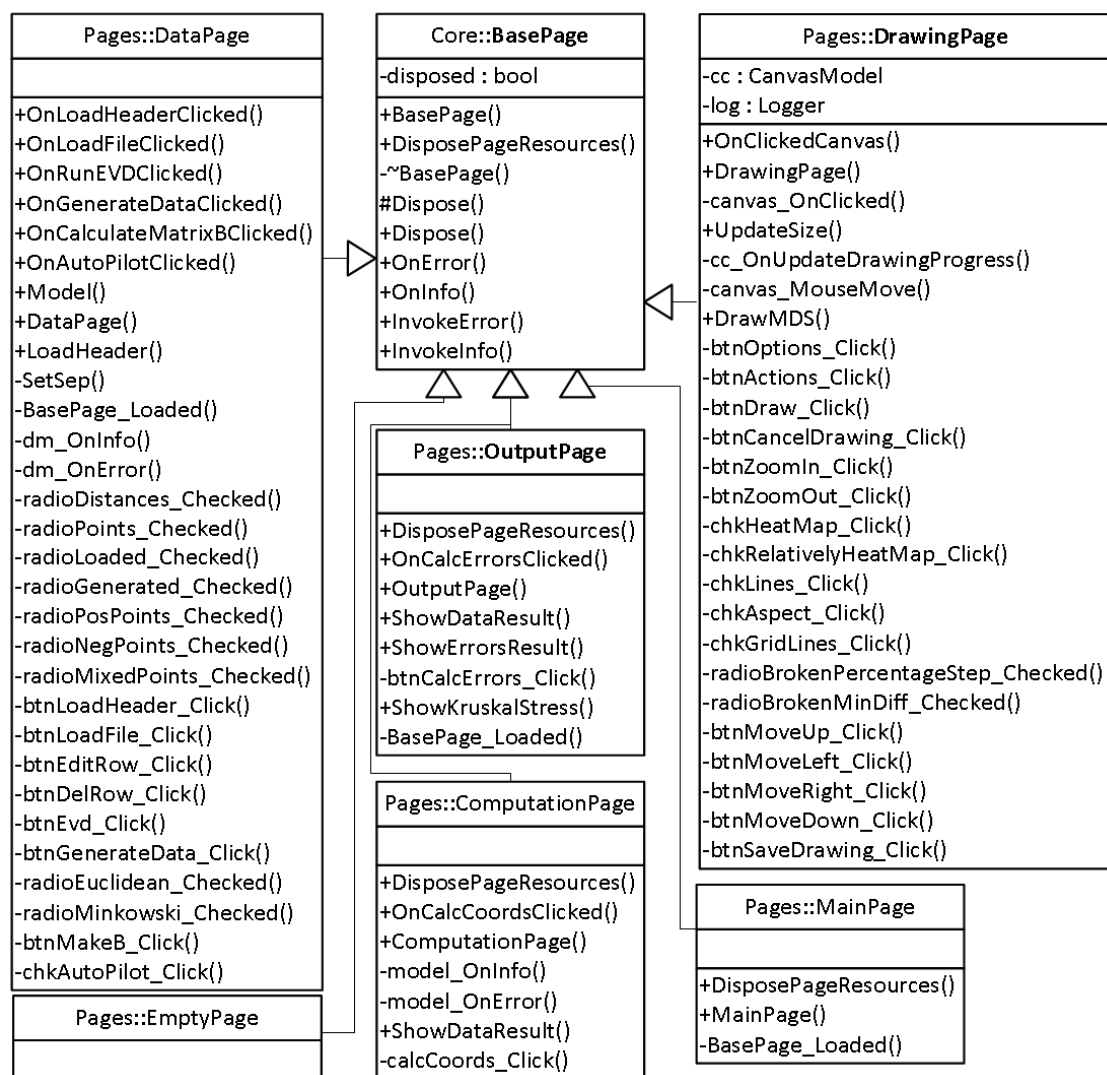
Na rysunku 7.1 znajduje się model ogólny, w którym linie pomarańczowe oznaczają zwykłą referencje, a linie fioletowe dziedziczenie.



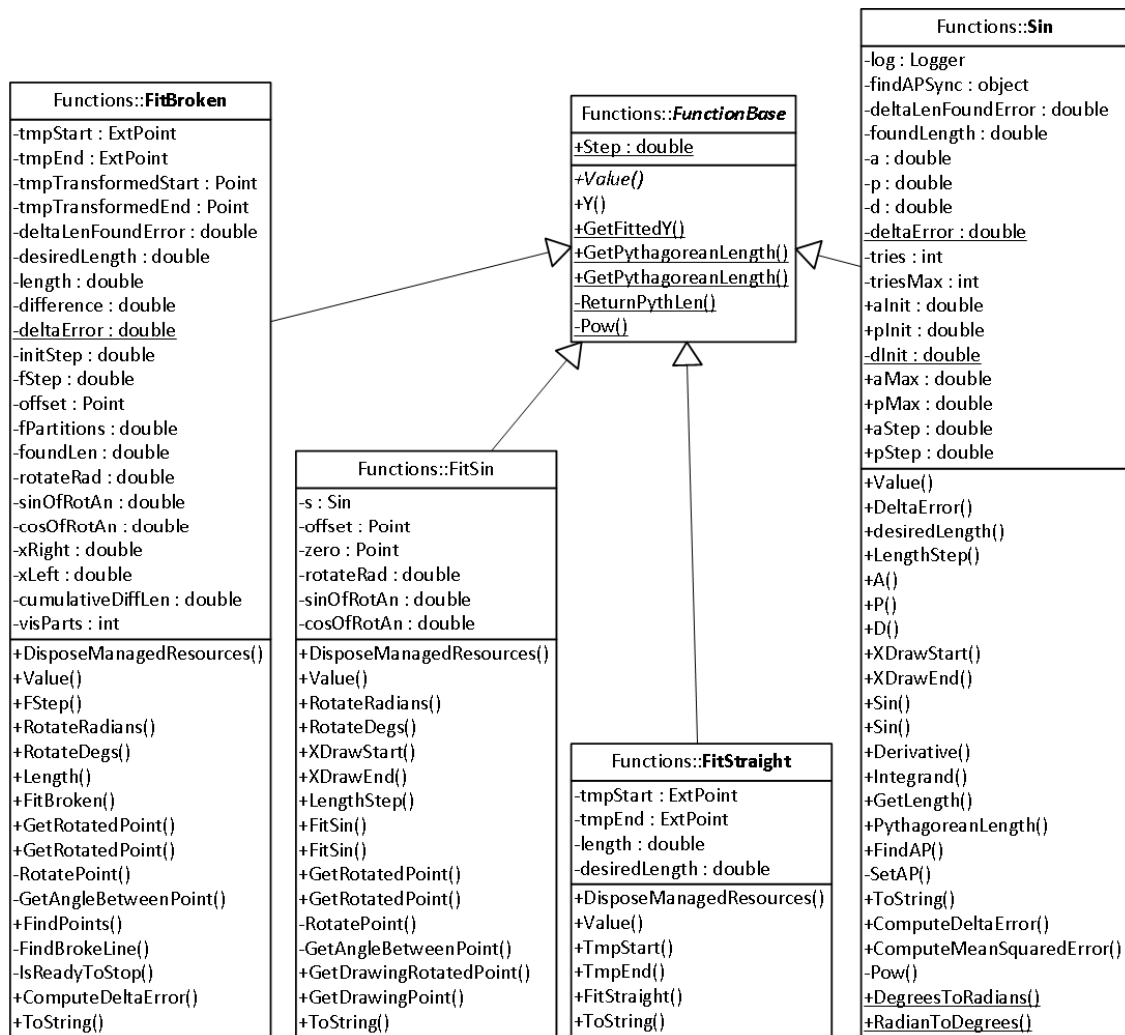
Rysunek 7.1 Model obiektowy głównych obiektów



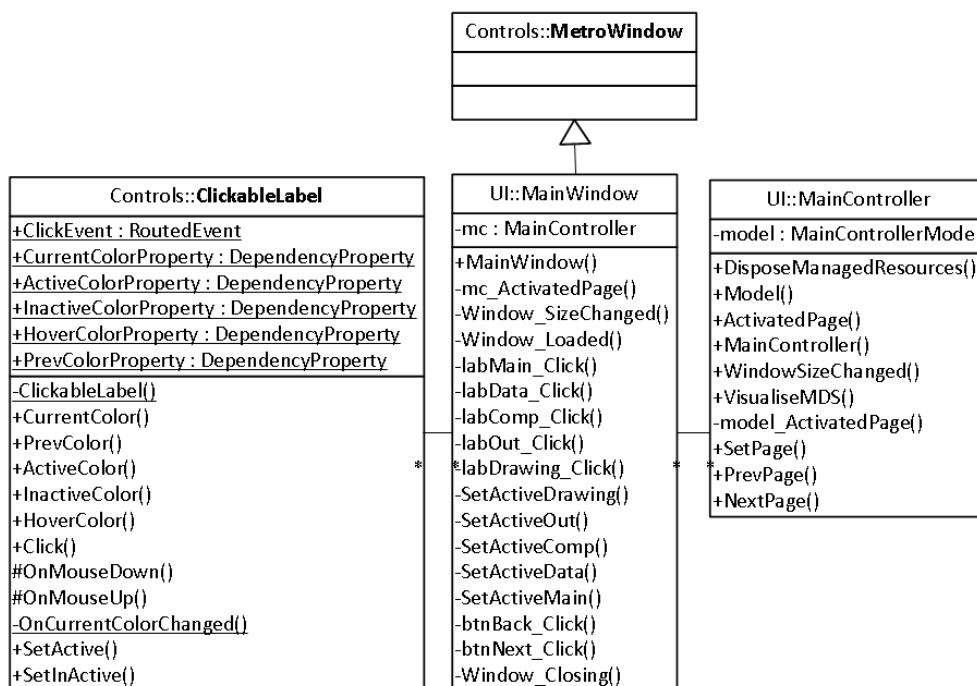
Rysunek 7.2 Model obiektowy modeli danych dla stron



Rysunek 7.3 Model obiektowy stron



Rysunek 7.4 Model obiektowy funkcji różnych typów linii

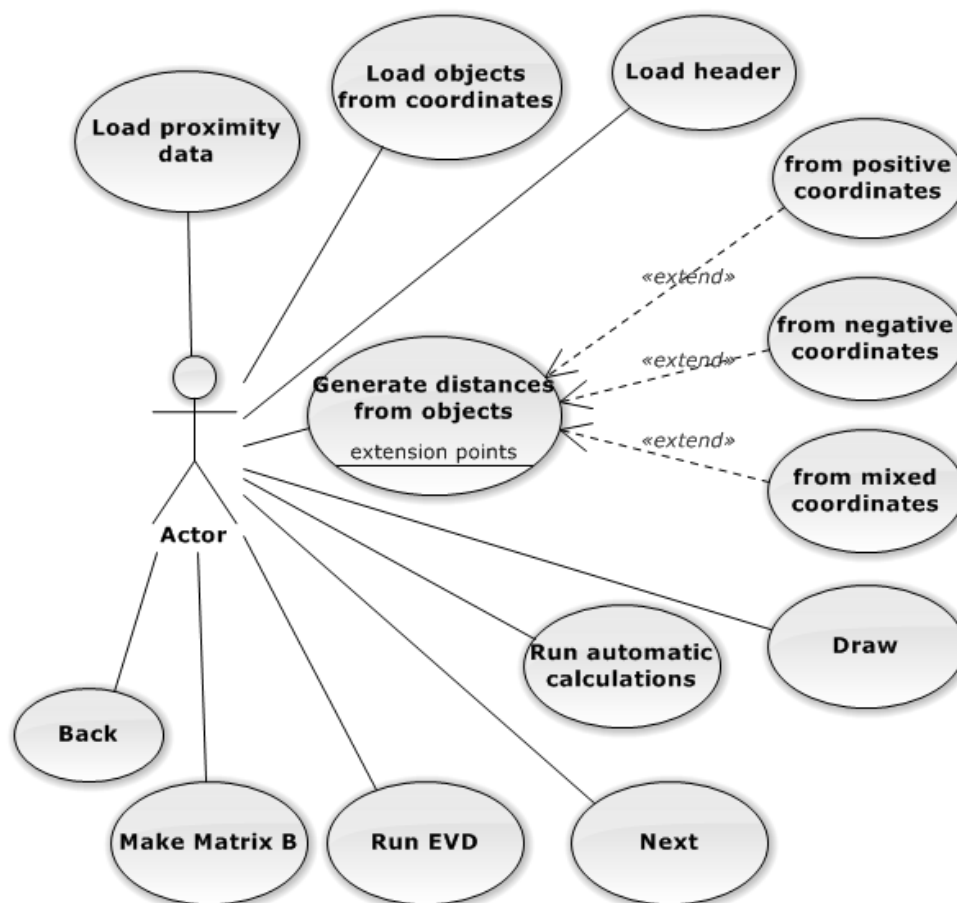


Rysunek 7.5 Model obiektowy głównego okna

7.4 Modele przypadków użycia

Rozdział ten został poświęcony opisowi najważniejszych modeli przypadków użycia, czyli dla modułu ładowania danych oraz wizualizacji.

7.4.1 Proces obsługi danych



Rysunek 7.6 Przypadki użycia podczas obsługi danych

Tabela 7.1 Opis tabelaryczny dla przypadków użycia podczas obsługi danych

Nazwa	Obsługa danych.
Powiązanie wymagania	Plik z danymi.
Kontekst zadaniowy	Użytkownik przechodzi na stronę „Data”.
Warunek pomyślnego zakończenia	System wyświetlił użytkownikowi macierz odległości. Po czym użytkownik wykonał obliczenia lub zaznaczył automatyczne obliczanie algorytmu MDS.
Warunek niepomyślnego zakończenia	1. Użytkownik załadował plik ze złymi danymi. 2. Użytkownik pominął obliczenia.
Aktorzy główni	Użytkownik.
Aktorzy drugoplanowi	System.
Wyzwalacz	Wywołanie strony „Data”.



Główny przepływ wykonania	Krok	Akcja
	1.	Użytkownik ładuje dane macierzy odległości z pliku.
	2.	System wyświetla dane w tabelce.
	3.	Użytkownik wykonuje obliczenia.
Rozszerzenia	Krok	Akcja
	1.1	Użytkownik generuje macierz odległości z obiektów.
	1.1.1	Wybiera obiekty ze współrzędnych nieujemnych.
	1.1.2	Wybiera obiekty ze współrzędnych ujemnych.
	1.1.3	Wybiera obiekty ze współrzędnych pozytywnych.
	1.2	Użytkownik ładuje nagłówek dla danych.
	3.1	Użytkownik wykonuje ręcznie kroki algorytmu MDS.
	3.1.1	Użytkownik oblicza macierz B.
	3.1.2	Użytkownik wywołuje algorytm EVD.
	3.1.3	System przenosi użytkownika na kolejny ekran.
	3.2	Użytkownik zaznacza opcję automatycznych obliczeń.
	3.2.1	Użytkownik przechodzi do końcowej akcji „Draw”.

7.4.2 Proces wizualizacji



Rysunek 7.7 Przypadki użycia podczas obsługi wizualizacji

Tabela 7.2 Opis tabelaryczny dla przypadków użycia podczas obsługi wizualizacji

Nazwa	Obsługa wizualizacji.
Powiązanie wymagania	Poprawne załadowanie danych. Obliczony wynik metody MDS.
Kontekst zadaniowy	Użytkownik przechodzi na stronę „Drawing”.
Warunek pomyślnego zakończenia	System zakończył rysowanie obiektów oraz poszczególnych linii między nimi.
Warunek niepomyślnego zakończenia	1. Użytkownik nie wykonał obliczeń metody MDS.
Aktorzy główni	Użytkownik.
Aktorzy drugoplanowi	System.
Wyzwalacz	Wywołanie strony „Drawing”.



Główny przepływ wykonania	Krok	Akcja
	1.	Użytkownik oczekuje na zakończenie rysowania.
	2.	System wyświetla wizualizację.
	3.	Użytkownik wykonuje operacje na wizualizacji.
Rozszerzenia	Krok	Akcja
	1.1	Użytkownik może zatrzymać rysowanie.
	1.2	Użytkownik może ponownie uruchomić wizualizację.
	1.3	Użytkownik może zaznaczyć dowolny obiekt.
	1.4	Użytkownik może zaznaczyć dowolną linię.
	3.1	Użytkownik może przemieścić wizualizację.
	3.1.1	Przesunięcie w górę.
	3.1.2	Przesunięcie w dół.
	3.1.3	Przesunięcie w prawo.
	3.1.4	Przesunięcie w lewo
	3.2	Użytkownik może powiększyć lub zmniejszyć wizualizację.
	3.3	Użytkownik może zapisać wynik wizualizacji do pliku.
	3.4	Użytkownik może powrócić do poprzedniego ekranu.
	3.5	Użytkownik może włączyć/wyłączyć kolorowanie linii.
	3.5.1	Użytkownik może włączyć/wyłączyć kolorowanie linii relatywnie.
	3.6	Użytkownik może wyłączyć/włączyć rysowanie linii między obiektami.
	3.7	Użytkownik może wyłączyć/włączyć rysowanie pomocniczych linii.
	3.8	Użytkownik może wyłączyć/włączyć aspekt osi współrzędnych.
	3.9	Użytkownik może ustawić częstotliwość rysowania linii pomocniczych oraz wartości na osiach współrzędnych.
	4.1	Użytkownik może ustawić algorytm rysowania linii przerywanych.
	4.1.1	Użytkownik może ustawić algorytm „minimum”.
	4.1.2	Użytkownik może ustawić algorytm „percentage”.



7.5 Algorytmy

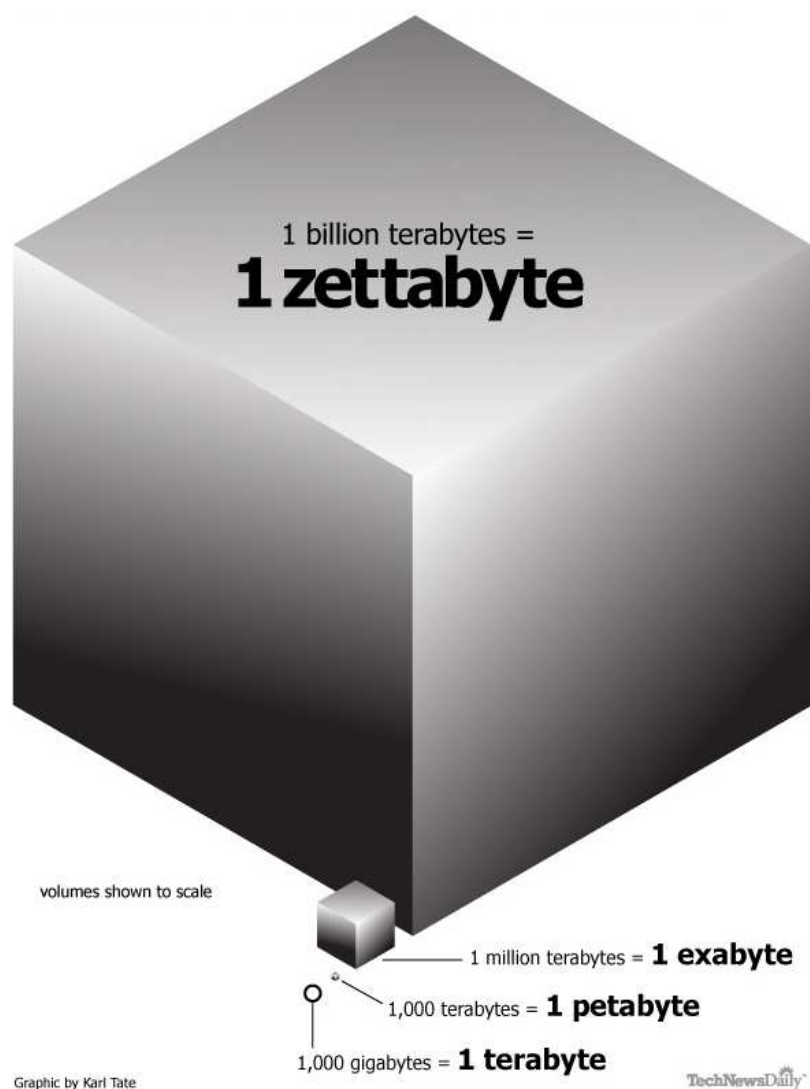
Tabela 7.3 Pseudokod algorytmu „minimum”

```
dla każdego (xLewy = środkowyPunkt.X; (xLewy - minKrok) >= 0; xLewy -= minKrok) {
    ( jeśli ilośćPodziałów > 0 ) {
        jeśli ( (xLewy == środkowyPunkt.X || ilośćPodziałów % 2 == 0) && (xLewy - resztaDoWypełnienia <= 0) ) {
            poprzedniLewyX = xLewy - minKrok;
            DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(poprzedniLewyX));
            DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(xLewy));
            zróbLewąPrzerwęWLinii = true;
        }
        jeśli ( (ilośćPodziałów <= 0 && !znaleziono) || (xLewy - resztaDoWypełnienia <= 0) ) {
            zróbLewąPrzerwęWLinii = true;
            zróbPrzerwyCiągłe = true;
        }
        jeśli ( zróbLewąPrzerwęWLinii ) {
            sumaWyrysowanychRóżnic += minKrok;
            zróbLewąPrzerwęWLinii = false;
        }
        jeśli ( ((długość_między_punktami - sumaWyrysowanychRóżnic) - oczekiwana_długość) <= marginesBłędu )
            znaleziono = true;
        else {
            jeśli (!zróbPrzerwyCiągłe) {
                DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(xPrawe));
                następnePraweX = xPrawe + minKrok;
                DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(następnePraweX));
            }
            zróbPrawąPrzerwęWLinii = true;
        }
    }
    jeśli ( zróbPrawąPrzerwęWLinii ) {
        sumaWyrysowanychRóżnic += minKrok;
        jeśli ( ((długość_między_punktami - sumaWyrysowanychRóżnic) - oczekiwana_długość) <= marginesBłędu )
            znaleziono = true;
        zróbPrawąPrzerwęWLinii = false;
    }
    ilośćPodziałów--;
} else { break; }
if (!znaleziono) xPrawe += minKrok;
}
```

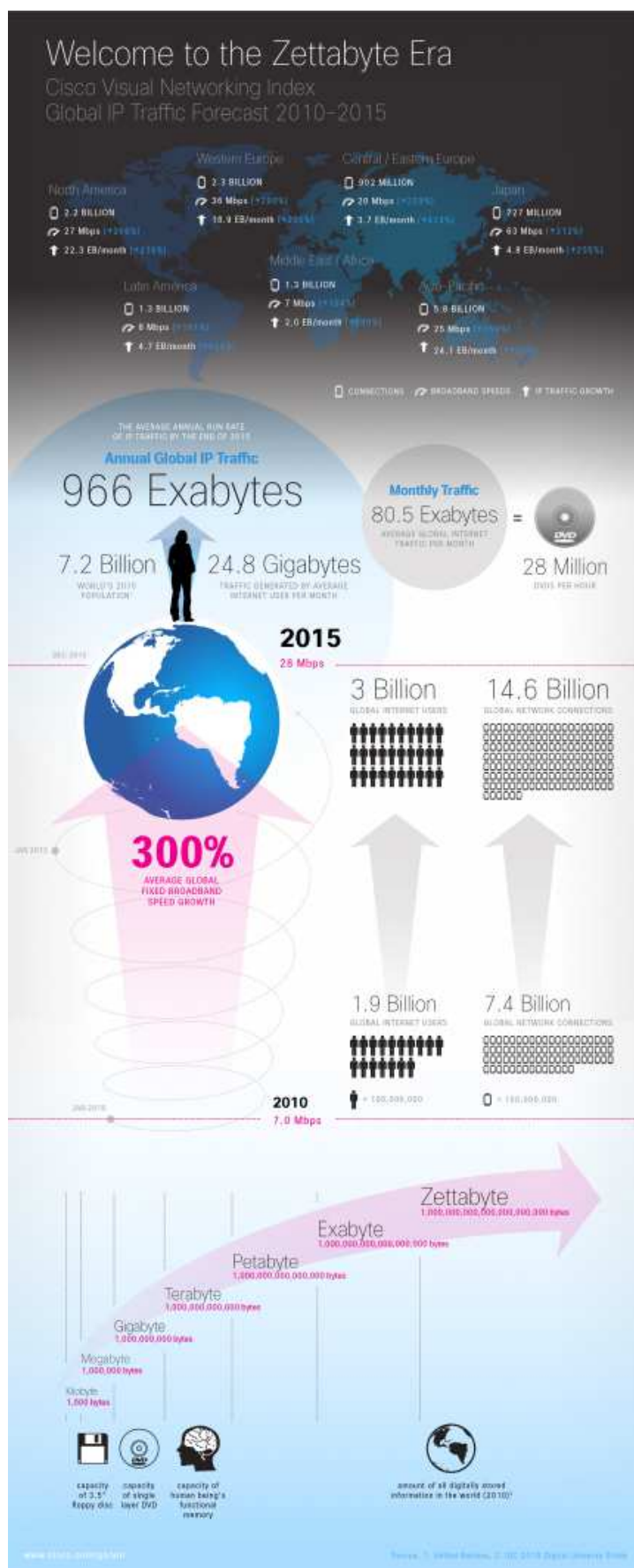
```
jeśli ( !(xLewy - resztaDoWypełnienia <= 0) ) {
    jeśli ( xLewy > 0 ) {
        DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(xLewy));
        DodajPunktDoRysowaniaLinii(0);
    }
    jeśli ( xPrawe < punkt_koncowy.X ) {
        DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(xPrawe));
        DodajPunktDoRysowaniaLinii(PobierzObróconyPunkt(punkt_koncowy.X));
    }
}
```



7.6 Dodatkowe ilustracje



Rysunek 7.8 Infografika ukazująca skalę jednostki zettabajt [TATE 2010]



Rysunek 7.9 Pełna wersja infografiki ukazującej skalę jednostki [WEBS 2010]