Uniwersytet Śląski
Wydział Nauk Ścisłych i Technicznych
Instytut Informatyki

Autoreferat rozprawy doktorskiej

Ewolucyjna agregacja rang w systemach rekomendacyjnych

Autor:
mgr Michał Bałchanowski

Opiekun naukowy:
prof. dr hab. Urszula Boryczka

Sosnowiec, 2023 r.

1. Wstęp

Aby wspomóc użytkownika w procesie podejmowania decyzji, zasugerowano wykorzystanie systemów rekomendacyjnych, których celem jest zaproponowanie użytkownikowi pewnych produktów lub usług, które z dużym prawdopodobieństwem mogą go zainteresować. Nie jest to problem trywialny i od wielu lat prowadzone są prace badawcze w tej tematyce. Najistotniejszym wydarzeniem, które znacznie zwiększyło zainteresowanie tym zagadnieniem, był konkurs zorganizowany 2 października 2006 roku przez firmę Netflix, gdzie badaczom, którym udało się wystarczająco zwiększyć jakość generowanych rekomendacji, oferowano do wygrania 1 milion dolarów. Spowodowało to, że na przestrzeni ostatnich lat powstało wiele algorytmów rekomendacyjnych, choć w środowisku naukowym nadal nie ma konsensusu co do tego, które z tych technik są najlepsze, szczególnie w kontekście problemu Top-N rekomendacji (z ang. *Top-N recommendation problem*) [2].

Celem zaproponowanego w rozprawie algorytmu, będzie próba przewidzenia listy przedmiotów (rankingu), który z dużym prawdopodobieństwem zainteresuje użytkownika. Do ewaluacji zaproponowanego podejścia wykorzystane zostaną stosowne miary, które uwzględniają pozycję relewantnych przedmiotów na rekomendowanej liście. Wykorzystanie tych miar do bezpośredniej optymalizacji rankingu nie jest łatwe, ponieważ posiadają one specyficzne właściwości, które uniemożliwiają zastosowanie klasycznych algorytmów aproksymacyjnych [8]. Z tego względu w rozprawie zaproponowany algorytm będzie bazował na algorytmie ewolucji różnicowej (z ang. *differential evolution*, DE), który umożliwi bezpośrednią optymalizację rankingu za pomocą funkcji rangującej.

W pracy poruszony zostanie również problem agregacji rang (z ang. *rank aggregation problem*) [12]. Jest to stosunkowo nowe podejście w kontekście systemów rekomendacji, gdzie zamiast jednego algorytmu wykorzystuje się pewien zbiór algorytmów, które generują rekomendacje dla danego użytkownika, a następnie wyniki tych algorytmów są agregowane w celu utworzenia nowej rekomendacji. Agregacja nie jest problemem trywialnym, ponieważ nie istnieje jedna, uniwersalna metoda, łączenia takich rankingów ze sobą. Szczegółowe motywacje, które skłoniły autora do podjęcia się tej tematyki, zostały opisane w podrozdziale 1.1.

1.1. Motywacja

W literaturze zaproponowano wiele algorytmów rekomendacyjnych, których celem jest zaproponowanie użytkownikowi pewnej uporządkowanej listy przedmiotów [2]. W tej pracy taką listę będziemy nazywali rankingiem. Podczas ewaluacji systemów rekomendacji, zazwyczaj generuje się rekomendacje dla poszczególnych użytkowników, następnie oblicza się jakość rekomendacji zgodnie z przyjętą miarą, a otrzymane wyniki uśrednia. Okazuje się jednak, że jeżeli porównamy zaproponowane rekomendacje w kontekście konkretnego użytkownika, to poszczególne algorytmy generują rekomendacje różniące się od siebie, a tym samym różnej jakości. W tabeli 1 zaprezentowany został prosty przykład ilustrujący ten problem.

Tabela 1: Trywialny przykład reprezentujący jak różne algorytmy rekomendacyjne, mogą generować rekomendacje różnej jakości, dla poszczególnych użytkowników. Jakość rekomendacji jest wyrażona przez wartość z przedziału od 0 do 1

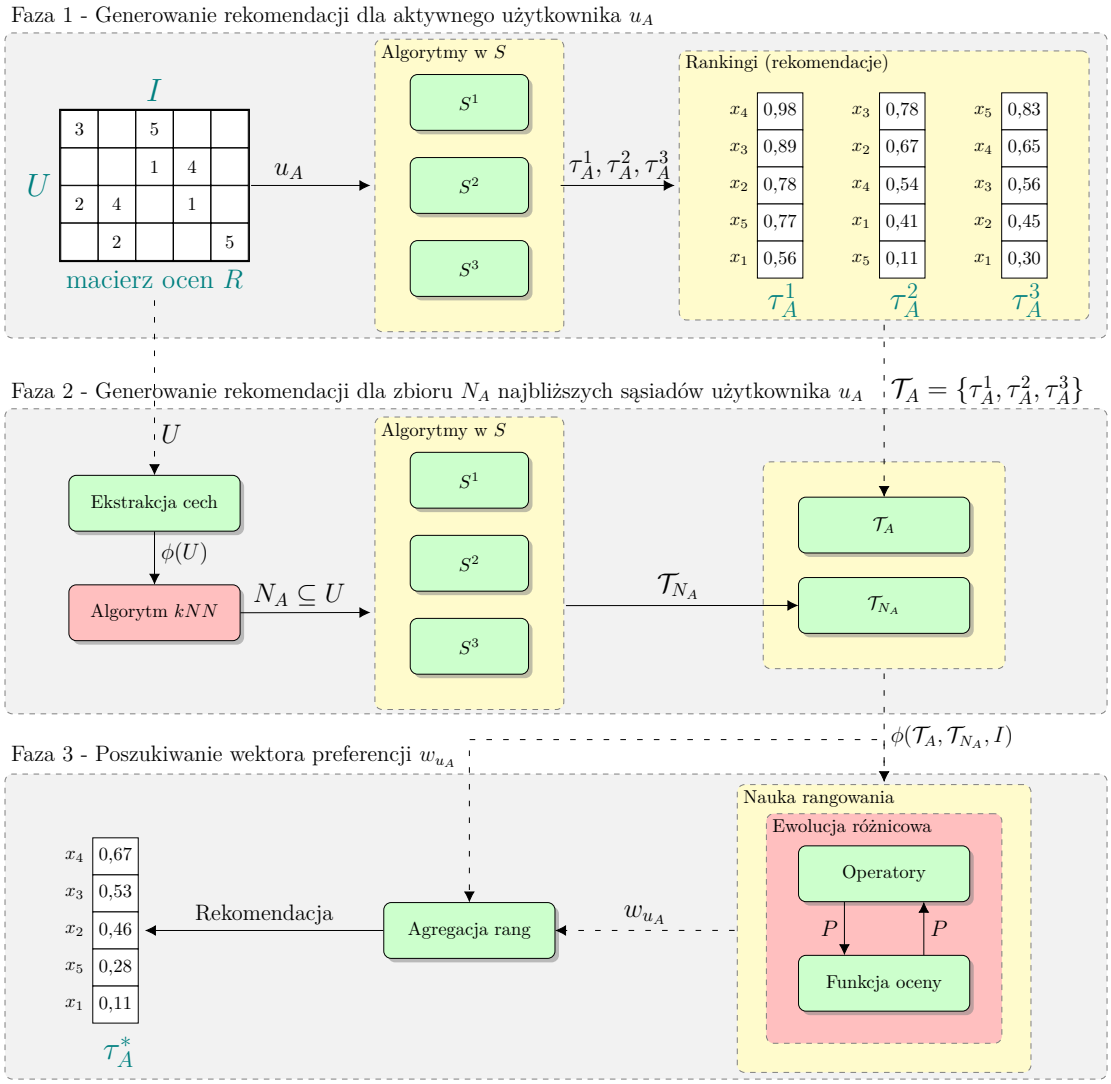
	Algorytm 1	Algorytm 2	Algorytm 3	Algorytm 4
użytkownik 1	0,5	0	0	1
użytkownik 2	0	0,5	1	0,5
użytkownik 3	1	0,5	1	0,5
użytkownik 4	0,5	1	0	0
średnia	0,5	0,5	0,5	0,5

Analizując tabelę 1 można zauważyć, że choć skuteczność algorytmów po uśrednieniu jest identyczna, to na poziomie poszczególnych użytkowników jakość rekomendacji jest różna. Ze względu na to, że nie ma jednego, uniwersalnego algorytmu, który generowałby rekomendacje wysokiej jakości dla wszystkich użytkowników w systemie, to w celu poprawienia jakości końcowej rekomendacji można wykorzystać techniki agregujące, które z powodzeniem były wykorzystywane choćby w systemach wyszukiwania informacji [12]. Umożliwiają one złączenie rekomendacji (rankingów), które zostały wygenerowane dla danego użytkownika, co w teorii powinno poprawić końcową jakość rekomendacji.

Wykorzystanie tej idei w systemach rekomendacji zostało już zasugerowane przez innych badaczy. W pracy [1, s. 417] autor zaproponował wykorzystanie algorytmów agregujących, w celu zwiększenia jakości generowanych rekomendacji. Wskazał, że jest to idea zaczerpnięta z klasyfikatorów złożonych [19], gdzie łączy się ze sobą kilka klasyfikatorów w celu utworzenia optymalnego modelu predykcyjnego. Dodatkowo autor zauważa, że jest to zagadnienie, które w kontekście systemów rekomendacyjnych nie zostało dobrze zbadane i jest ciekawym kierunkiem przyszłych prac badawczych.

Podobne wnioski zostały przedstawione w pracy [20], gdzie autorzy wskazali, że choć w literaturze zaproponowano wiele algorytmów rekomendacyjnych, to często generowane przez nie rekomendacje różnią się od siebie, co stwarza okazję do poprawy końcowej rekomendacji, poprzez agregację wyników różnych algorytmów. Autorzy wskazują również, że w kontekście systemów rekomendacyjnych powstało stosunkowo mało dedykowanych algorytmów, które odnosiłyby się do tego problemu.

Z tego względu autor dostrzega w tym obszarze **lukę badawczą**, którą postara się wypełnić, tym samym tworząc podstawę pod kolejne prace badawcze w tej tematyce. Na rysunku 1 znajduje się diagram, prezentujący ogólną ideę zaproponowanego podejścia.



Rysunek 1: Diagram przestawiający ogólną ideę algorytmu, zaproponowanego w rozprawie. Kolorem czerwonym oznaczono główne obszary badawcze. Objaśnienie poszczególnych symboli znajduje się w tabeli 6

1.2. Teza

Zaproponowany algorytm ewolucyjnej agregacji rang, poprawia jakość generowanej agregacji, w porównaniu do wybranych metod zaproponowanych w literaturze.

Pod pojęciami:

- *Jakość generowanej agregacji* – należy rozumieć jakość rekomendacji, wygenerowaną na podstawie tej agregacji, wyrażonej przy pomocy miary uśrednionej średnej precyzji (z ang. *mean average precision*, MAP), która została opisana w rozdziale 2.
- *Wybranych metod* – należy rozumieć pięć metod pozycyjnych, które zostały przedstawione w rozdziale 4.

1.3. Cele rozprawy

Głównym celem rozprawy jest opracowanie autorskiego algorytmu agregacji, dostosowanego do wykorzystania w systemach rekomendacji i bazującym na algorytmie DE. Zadaniem tego algorytmu, będzie agregacja rankingów wygenerowanych przez poszczególne algorytmy rekomendacyjne i zaprezentowanie użytkownikowi finalnego rankingu, który będzie dostosowany do jego personalnych preferencji. Algorytm ten zostanie porównany z innymi technikami wykorzystywanymi do tworzenia agregacji. Kolejne etapy realizacji głównego celu pracy obejmują:

1. **Przegląd literatury związanej z tematyką rozprawy**. W przeglądzie literaturowym znajdą się ogólne informacje na temat systemów rekomendacji oraz przedstawiony zostanie problem agregacji rang. Dodatkowo zaprezentowany zostanie problem uczenia się rankingu oraz algorytm DE.

2. **Zaprezentowanie zbioru danych wykorzystanego do przeprowadzenia eksperymentów**. Badanie skuteczności zaproponowanego podejścia będzie realizowane na rzeczywistym zbiorze danych, który jest ogólnodostępny i bardzo popularny w środowisku badawczym zajmującym się systemami rekomendacyjnymi. W pracy zaprezentowana zostanie również krótka analiza tego zbioru, żeby przybliżyć czytelnikowi jego zawartość.

3. **Dostrojenie parametrów poszczególnych algorytmów wchodzących w skład agregacji**. Każdy z algorytmów, który generuje rekomendacje posiada swoje parametry, które wymagają odpowiedniego dostrojenia przed przystąpieniem do procesu rekomendacji. Przedstawiony zostanie przykładowy wpływ poszczególnych parametrów na uzyskiwane wyniki.

4. **Włączenie do procesu agregacji rankingów innych użytkowników w systemie**. W celu poprawy jakości generowanych rekomendacji, zaprezentowana zostanie modyfikacja, która uwzględnia w tym procesie rankingi innych użytkowników.

5. **Badanie wpływu różnych wariantów funkcji oceny, na jakość agregacji**. Funkcja oceny jest kluczowym elementem każdego algorytmu metaheurystycznego. Z tego względu przetestowane zostaną różne warianty tej funkcji, w celu zbadania wpływu jej poszczególnych wariantów na jakość generowanych rekomendacji.

2. Miary jakości rekomendacji

W celu oceny jakości generowanych rekomendacji, wykorzystywane są różne metryki, których zdaniem jest obliczenie jak dobrze system potrafi przewidzieć preferencje aktywnego użytkownika. Precyzja i czułość są to dwie klasyczne miary, które często są wykorzystywane przy ocenianiu jakości klasyfikacji. W kontekście systemów rekomendacji, precyzja, reprezentuje procent przedmiotów relewantnych, które pojawiły się w zarekomendowanej liście, natomiast czułość reprezentuje procent relewantnych przedmiotów, które zostały zarekomendowane. Oblicza się je według następujących wzorów:

$$precyzja = 100 \cdot \frac{|\{zarekomendowane\ przedmioty\} \cap \{relewantne\ przedmioty\}\}|}{|\{zarekomendowane\ przedmioty\}|}, \quad (1)$$

$$czulosc = 100 \cdot \frac{|\{zarekomendowane\ przedmioty\} \cap \{relewantne\ przedmioty\}\}|}{|\{relewantne\ przedmioty\}|}, \quad (2)$$

gdzie *zarekomendowane przedmioty* są to przedmioty, które zostały zaproponowane przez system rekomendacyjny, natomiast *relewantne przedmioty* są to przedmioty, które znajdują się w zbiorze testowym aktywnego użytkownika.

Zasadniczą wadą zwykłej precyzji jest to, że nie bierze ona pod uwagę pozycji elementów relewantnych w jakiej znajdują się one na liście. Z tego względu do oceny jakości rekomendacji, częściej stosuje się miarę średniej precyzji (z ang. *average precision*, AP), która uśrednia wartości precyzji obliczanej dla poszczególnych pozycji w zarekomendowanej liście. Średnią precyzję definiujemy zgodnie ze wzorem:

$$AP@K = \frac{\sum_{k=1}^{K} (precyzja@k \cdot rel(k))}{\min(|\{relewantne\ przedmioty\}|, K)}, \quad (3)$$

gdzie K oznacza rozmiar zarekomendowanej listy, a $rel(k)$ jest to funkcja charakterystyczna, która przyporządkowuje wartość 1 w przypadku gdy przedmiot na pozycji k jest relewantny (w przeciwnym wypadku 0). Należy zauważyć, że w systemie rekomendacyjnym może występować więcej przedmiotów niż te, które są prezentowane użytkownikowi. Z tego względu przy obliczeniu tej miary rozpatruje się tylko K pierwszych przedmiotów, co jest oznaczane jako $@K$ (z ang. *cutoff-treshold*). Zaletą tej miary jest to, że penalizuje ona niepoprawne uporządkowanie przedmiotów na liście. Średnia precyzja opisana powyżej, zazwyczaj wykorzystywana jest podczas ewaluacji rekomendacji w kontekście jednego użytkownika. Jednak często jako wynik działania algorytmu chcemy otrzymać jedną liczbę, która określi jakość działania algorytmu. Z tego względu zaproponowano uśrednioną średnią precyzję (z ang. *mean average precision*, MAP) wyrażoną następującym wzorem:

$$MAP@K = \frac{\sum_{u=1}^{U} (AP@K)_u}{|U|}, \quad (4)$$

gdzie U oznacza zbiór użytkowników. Miary AP i MAP często są wykorzystywane przy ocenach binarnych. W sytuacji kiedy w systemie występują różne poziomy istotności i posiadamy informację na temat tego, jak bardzo dany przedmiot jest relewantny (np. w skali od 1 do 5), to warto wykorzystać miarę znormalizowanego zdyskontowanego skumulowanego zysku (z ang. *normalized discounted cumulative gain*, NDCG). Podobnie jak w przypadku miary MAP, celem tej miary jest premiowanie przedmiotów, które znajdują się wysoko (bliżej pierwszej pozycji) na rekomendowanej liście. Aby lepiej zrozumieć sposób obliczania tej miary, w pierwszej kolejności należy zwrócić uwagę na sposób obliczania miary skumulowanego zysku (z ang. *cumulative gain*, CG), która wyrażona jest następującym wzorem:

$$CG_K = \sum_{k=1}^{K} rel_k, \tag{5}$$

gdzie K oznacza liczbę przedmiotów znajdującą się na rekomendowanej liście, natomiast rel_k oznacza stopień relewantności danego przedmiotu na pozycji k. Niestety zasadniczą wadą miary CG jest to, że nie uwzględnia ona pozycji relewantnych przedmiotów. Z tego względu zaproponowano zdyskontowany skumulowany zysk (z ang. *discounted cumulative gain*, DCG) wyrażony następującym wzorem:

$$DCG_K = \sum_{k=1}^{K} \frac{rel_k}{log_2\,(k+1)}. \tag{6}$$

W mianowniku tego wzoru występuje logarytm, który penalizuje stopień relewantności proporcjonalnie do pozycji przedmiotu na rekomendowanej liście. Niestety miary DCG nie można porównywać pomiędzy użytkownikami, ze względu na to, że każdy z użytkowników będzie posiadać różną liczbę relewantnych przedmiotów w swoich rekomendacjach. Z tego względu należy dokonać normalizacji i do tego celu wykorzystuje się idealny (wzorcowy) zdyskontowany skumulowany zysk (z ang. *ideal discounted cumulative gain*, IDCG), który wykorzystywany jest jako współczynnik normalizujący, a wyrażony jest następującym wzorem:

$$IDCG_K = \sum_{k=1}^{|REL_K|} \frac{rel_k}{log_2\,(k+1)}, \tag{7}$$

gdzie REL_K reprezentuje listę wszystkich relewantnych przedmiotów dla danego użytkownika, aż do pozycji K, które są posortowane zgodnie z ich poziomem istotności. Następnie aby uzyskać miarę NDCG, należy:

$$NDCG_K = \frac{DCG_K}{IDCG_K}. \tag{8}$$

3. Ewolucja różnicowa

Ewolucja różnicowa jest to metaheurystyka, która została opracowana przez K. Price'a i R. Storna [22] w 1997 roku. Znalazła ona zastosowanie w wielu zagadnieniach i problemach optymalizacyjnych, choć zazwyczaj jest wykorzystywana do optymalizacji problemów ciągłych. Podobnie jak w klasycznych algorytmach ewolucyjnych występuje tutaj pewna populacja, która inicjalizowana jest poprzez losowe rozmieszczenie osobników w przestrzeni rozwiązań. Formalnie populację P będziemy zapisywać jako:

$$P = \{w_1, w_2, \ldots, w_{NP}\}, \tag{9}$$

gdzie:

NP – liczba osobników w populacji.

Każdy osobnik w populacji P jest zazwyczaj reprezentowany przez pewien wektor liczb rzeczywistych, który reprezentuje rozwiązanie danego problemu optymalizacyjnego:

$$w_g = \{w_{g,1}, w_{g,2}, \ldots, w_{g,b}\}, \tag{10}$$

gdzie:

b – jest to wymiar wektora.

W algorytmie ewolucji różnicowej wyróżniamy dwa operatory: mutacja i krzyżowanie. Ich celem jest ciągła zmiana osobników podczas procesu ewolucji, aż do osiągnięcia kryterium stopu. W podstawowej wersji algorytmu wariant mutacji, który w literaturze nazywany jest wariantem *DE/rand/1*, można wyrazić następującym wzorem:

$$v_i = w_{r_1} + F\left(w_{r_2} - w_{r_3}\right), \tag{11}$$

gdzie:

v_i – jest to nowy wektor,

r_1, r_2, r_3 – są to trzy losowe numery osobników z populacji P, przy czym $r_1 \neq r_2 \neq r_3$,

F – jest współczynnikiem wzmocnienia i przyjmuje wartość z przedziału $[0, 1]$.

Kolejnym etapem algorytmu jest zastosowanie operatora krzyżowania, który tworzy nowego osobnika z_i, poprzez połączenie genotypów rodzica w_i z populacji rodziców P, oraz osobnika v_i powstałego w wyniku zastosowania operatora mutacji:

$$z_{i,j} = \begin{cases} v_{i,j} & \text{gdy } (rand(j) \leq CR \text{ lub } i = i_{rand}) \\ w_{i,j} & \text{w przeciwnym wypadku.} \end{cases} \tag{12}$$

gdzie CR jest to prawdopodobieństwo krzyżowania, a i_{rand} to losowa liczba ze zbioru $\{1, 2, \ldots, NP\}$. Algorytm 2 przedstawia pseudokod algorytmu DE.

Algorytm 2 Pseudokod algorytmu ewolucji różnicowej

Wejście: **NP** – liczba osobników w populacji, **F** – współczynnik wzmocnienia, **CR** – prawdopodobieństwo krzyżowania, **b** – wymiar wektora
Wyjście: w_{best} – najlepszy osobnik z populacji P

Zainicjuj populację P o rozmiarze **NP**
Zainicjuj populację próbną Z o rozmiarze **NP**

repeat
    for $i := 1$ to **NP do**
        wygeneruj trzy losowe liczby r_1, r_2, $r_3 \in \{1, 2, \ldots, \mathbf{NP}\}$, gdzie $r_1 \neq r_2 \neq r_3 \neq i$
        $v_i = w_{r_1} + \mathbf{F}\,(w_{r_2} - w_{r_3})$ ▷ nowy osobnik v_i

        for $j := 1$ to **b do**
            wygeneruj losową liczbę $j_{rand} \in \{1, 2, \ldots, \mathbf{NP}\}$
            $z_{i,j} = \begin{cases} v_{i,j} & \text{gdy } (rand(0,1) \leq \mathbf{CR} \text{ lub } j = j_{rand}) \\ w_{i,j} & \text{w przeciwnym wypadku.} \end{cases}$
        end for

        if fitness(z_i) $\leq$ fitness(w_i) **then** ▷ warunek w zależności od kryterium
            umieść osobnika z_i w populacji próbnej Z
        else
            umieść osobnika w_i w populacji próbnej Z
        end if
    end for

    $P := Z$
until warunek stopu

oceń_wszystkie_osobniki_w_populacji(P)
w_{best} := wybierz_najlepszego_osobnika(P)
return w_{best}

4. Agregacja rang

Problem agregacji rang (z ang. *rank aggregation*), w literaturze nazywanym również agregacją preferencji (z ang. *aggregation of preferences*) [9], odnosi się do sytuacji, w której mając do dyspozycji kilka rankingów, naszym zadaniem jest utworzenie nowego rankingu, który będzie *lepszy* od rankingów bazowych.

Formalnie możemy to zapisać w następujący sposób. Załóżmy, że dysponujemy pewnym zbiorem elementów (obiektów, przedmiotów itp.) $I = \{x_1, x_2, \ldots, x_m\}$. Ranking definiujemy jako uporządkowaną listę tych elementów $\tau = [x_j \geq x_h \geq \cdots \geq x_z]$, gdzie $\geq$ oznacza relację porządku pomiędzy elementami ze zbioru I, a istotność danego elementu jest określona przez jego pozycję. Symbolem $\tau(x_j)$ oznaczana będzie pozycja (lub ranga) przedmiotu x_j w rankingu τ. Dwa przedmioty x_j i x_h można ze sobą porównać, wykorzystując ich pozycje w rankingu τ. Przykładowo możemy powiedzieć, że przedmiot x_j jest na *lepszej* pozycji w rankingu od przedmiotu x_h, co będzie oznaczane jako $\tau(x_j) < \tau(x_h)$. Dodatkowo pojedynczy algorytm będzie oznaczany jako S^r, a zbiór wszystkich algorytmów będzie oznaczany jako $S = \{S^1,\ S^2,\ ...,\ S^n\}$. Każdy z algorytmów w zbiorze S, generuje ranking τ^r, a zbiór wszystkich rankingów oznaczany będzie jako $\mathcal{T} = \{\tau^1, \tau^2, \ldots, \tau^n\}$, gdzie n oznacza liczbę algorytmów i liczbę wygenerowanych rankingów. Należy zauważyć, że występuje zależność pomiędzy liczbą algorytmów należących do zbioru S, a liczbą rankingów, które znajdują się w zbiorze $\mathcal{T}$.

Celem agregacji rang jest stworzenie nowego rankingu τ^*, który będzie *lepszy* niż poszczególne rankingi w zbiorze $\mathcal{T}$. Oczywiście sformułowanie *lepszy ranking* można interpretować na różne sposoby, a jego znaczenie należy rozpatrywać w kontekście danego problemu, mając na uwadze jego specyfikę. Przykładowo w systemach rekomendacji może to oznaczać taki ranking, który w największym stopniu podnosi jakość rekomendacji, gdzie jakość ta obliczana jest na podstawie miar opisanych w rozdziale 3.

Problem agregacji rang sprowadza się do zdefiniowania funkcji agregującej Ψ, która na podstawie rankingów w zbiorze $\mathcal{T}$, generuje nowy ranking τ^*:

$$\Psi : \{\tau^1, \tau^2, ..., \tau^n\} \rightarrow \tau^*. \tag{13}$$

W zależności od dostępnych danych, funkcja agregująca Ψ może zostać utworzona, opierając się na różnych metodach. Jedną z najpopularniejszych metod jest metoda Bordy (z ang. *Borda count*) [4], która każdemu elementowi $x_j \in I$ w rankingu τ^r przyporządkowuje wartość zgodnie ze wzorem:

$$x_{j\_borda} = \sum_{\tau^r | \forall \tau^r \in \mathcal{T}, x_j \in \tau^r} |\tau^r| - \tau^r(x_j) + 1, \tag{14}$$

po czym elementy te są sortowane w kolejności malejącej. Innymi zaproponowanymi w literaturze metodami agregującymi są metody zaliczane do rodziny metod *Comb** [15]. Metody te wykorzystują funkcję:

$$sc(x_j, \tau^r) = 1 - \frac{\tau^r(x_j) - 1}{|\tau^r|}, \tag{15}$$

która przyporządkowuje wartość każdemu przedmiotowi $x_j \in I$ w rankingu τ^r. Następnie rankingi są agregowane przy pomocy metod przedstawionych w tabeli 2.

Tabela 2: Wybrane algorytmy agregujące zaproponowane w [15]

Nazwa metody	Wzór
CombMIN	$\min_{\tau^r \in T} sc(x_j, \tau^r)$
CombMAX	$\max_{\tau^r \in T} sc(x_j, \tau^r)$
CombSUM	$\sum_{\tau^r \in T} sc(x_j, \tau^r)$
CombMED	$\frac{1}{\lvert T \rvert} CombSUM(x_j, \tau^r)$

5. Uczenie rangowania

W literaturze zaproponowano dedykowany typ algorytmów, który wykorzystywany jest do przewidzenia optymalnego uporządkowania elementów w rankingu. Podejście to nazywane jest *nauką rangowania* [14, s. 392]. Przedstawienie problemu optymalizacyjnego w ten sposób, szczególnie w kontekście systemów wyszukiwania informacji i rekomendacji jest uzasadnione, ponieważ wyniki działania tych systemów są często prezentowane użytkownikowi w formie uporządkowanej listy pewnych elementów. Ma to związek z tym, że na portalach internetowych mamy ograniczoną ilość miejsca i użytkownik z większym prawdopodobieństwem *skonsumuje* treści, które są mu zaprezentowane w pierwszej kolejności. Z tego względu najlepiej, żeby prezentowany ranking zawierał jak najwięcej relewantnych treści na początkowych pozycjach.

Nadzorowany proces nauki rangowania można w uproszczeniu podzielić na dwie fazy: treningową i testową. W fazie treningowej dostarczamy algorytmowi uczącemu pewnego zbioru zapytań, gdzie każdemu zapytaniu odpowiada pewien zbiór dokumentów. Dodatkowo każdy dokument ma przypisaną pewną etykietę (np. określoną przez człowieka), która określa jego relewantność w stosunku do danego zapytania. Celem fazy treningowej jest skonstruowanie modelu rangującego, który generalizuje wiedzę na podstawie dostępnych w zbiorze treningowym zapytań.

Po fazie treningowej następuje faza testowa, której celem jest sprawdzenie jakości utworzonego modelu. W tym celu wykorzystuje się pewien zbiór zapytań oraz dokumentów, który nie uczestniczył w procesie uczenia się i zwyczajowo nazywany jest zbiorem testowym. Choć przeważnie naukę rangowania wykorzystuje się w kontekście systemów wyszukiwania informacji, to jest to coraz bardziej popularne podejście również w kontekście systemów rekomendacji [18].

Formalnie problem uczenia się rangowania możemy zapisać w następujący sposób [3]. Przyjmijmy pewien zbiór zapytań $Q = \{q_1, \ldots, q_{|Q|}\}$ oraz pewien zbiór dokumentów $D = \{d_1, \ldots, d_{|D|}\}$. Zbiór treningowy jest utworzony na podstawie zbioru par zapytanie-dokument $(q_o, d_p) \in Q \times D$. Ponadto para zapytanie-dokument (q_o, d_p) jest reprezentowana przez wektor cech $\phi(q_o, d_p)$. Funkcję rangującą, która jest aproksymatorem liniowym [10, s. 441], zapisujemy w następujący sposób:

$$f(q_o, d_p) = w^T \phi(q_o, d_p), \tag{16}$$

gdzie w oznacza wektor, w którym poszczególne elementy tego wektora określają stopień istotności odpowiadającej cechy, w wektorze cech $\phi(q_o, d_p)$. W celu utworzenia rankingu dla danego zapytania q_o, obliczamy funkcję $f\,(q_o,\ d_p)$ dla każdego dokumentu d_p, używając do tego wzoru (16) i sortujemy te dokumenty w kolejności malejącej.

6. Zaproponowany algorytm EAR

Celem algorytmu *Ewolucyjnej Agregacji Rang* (EAR) zaproponowanego w rozprawie jest poprawa jakości generowanych rekomendacji, poprzez agregację rankingów, które zostały wygenerowane przez poszczególne algorytmy rekomendacyjne w zbiorze S. Takie podejście umożliwia utworzenie nowego rankingu τ_A^*, dopasowanego do preferencji aktywnego użytkownika u_A.

Algorytm EAR poszukuje wektora preferencji w_{u_A} dla aktywnego użytkownika u_A, poprzez optymalizację funkcji przystosowania na zbiorze treningowym TS. Wektor w_{u_A} określa stopień w jakim poszczególne algorytmy uczestniczą w procesie agregacji, ponieważ każdy element tego wektora jest przyporządkowany do jednego algorytmu. Wektor preferencji w_{u_A} wykorzystywany jest w funkcji rangującej wyrażonej wzorem:

$$f(u_A, x_j) = w_{u_A}^T \phi(u_A, x_j), \tag{17}$$

gdzie $\phi(u_A, x_j)$ jest to reprezentacja wektorowa przedmiotu $x_j \in I$ dla danego użytkownika u_A, gdzie elementy tego wektora dopowiadają wartościom przyporządkowanym przez poszczególne algorytmy rekomendacyjne ze zbioru S.

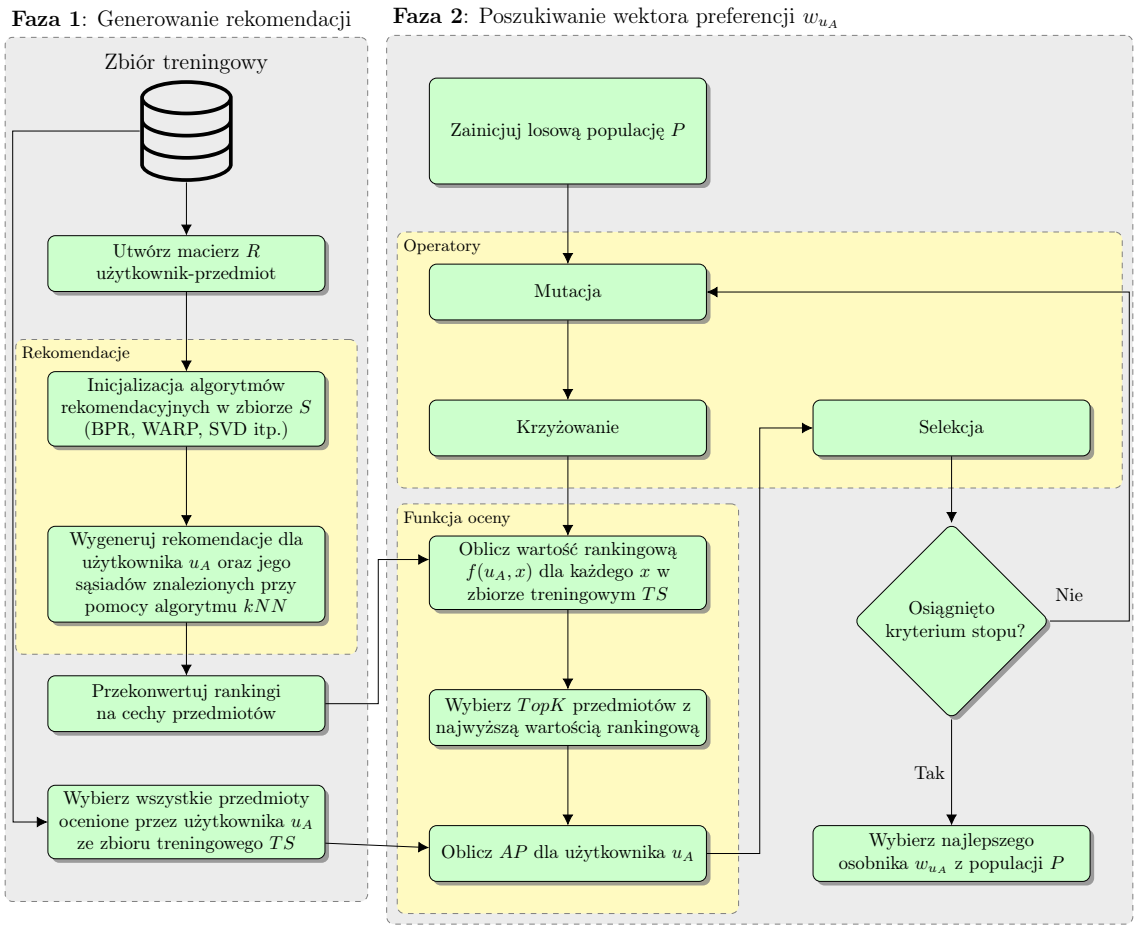
Należy pamiętać, że wartości tych cech będą inne dla każdego z użytkowników w systemie, bo dla każdego użytkownika algorytmy rekomendacyjne wygenerują inne rekomendacje. Z tego względu wektor w_{u_A} jest tworzony osobno dla każdego z użytkowników, ponieważ każdy z użytkowników posiada swoje indywidualne preferencje dotyczące rekomendacji.

Wykorzystanie algorytmu bazującego na mechanizmie ewolucji, wiąże się z wymogiem zdefiniowania funkcji przystosowania, nazywanej również funkcją oceny. Dzięki niej do kolejnych generacji będą mogły przechodzić osobniki najlepiej przystosowane, a cały algorytm będzie dążyć do znalezienia *optymalnego* rozwiązania. W przypadku algorytmu zaproponowanego w rozprawie, głównym elementem funkcji przystosowania będzie miara AP (wzór (3)), która jest obliczana dla aktywnego użytkownika u_A zgodnie z następującym wzorem:

$$Fitness(w_{u_A}, RT, ST) = AP@k(w_{u_A}, RT, ST), \tag{18}$$

gdzie ST jest to zbiór przedmiotów, które użytkownik u_A ocenił w swoim zbiorze treningowym TS, a RT jest to zbiór przedmiotów zarekomendowanych przez system. Formuła (18) oznacza średnią precyzję jaka zostałaby obliczona pomiędzy zbiorami RT oraz ST. Do wygenerowania zbioru RT wykorzystywany jest wektor wag w_{u_A} w funkcji rangującej (17), gdzie wektor ten jest pojedynczym osobnikiem w algorytmie DE.

Architektura zaproponowanego systemu rekomendacyjnego została przedstawiona na rysunku 2, zaś algorytm 3 przedstawia pseudokod algorytmu EAR.



Rysunek 2: Architektura systemu rekomendacyjnego. Proces rekomendacji jest podzielony na dwie fazy. W pierwszej fazie algorytmy rekomendacyjne generują rekomendacje w formie rankingów dla aktywnego użytkownika u_A. W drugiej fazie na bazie zbioru treningowego użytkownika u_A, algorytm DE wyszukuje optymalny wektor wag w_{u_A}

Algorytm 3 Pseudokod algorytmu Ewolucyjnej Agregacji Rang (EAR)

Wejście: **S** – zbiór algorytmów rekomendacyjnych, **U** – zbiór użytkowników, **I** – zbiór przedmiotów, **k** – parametr określający liczbę sąsiadów, **TS** – zbiór treningowy, $\boldsymbol{u_A}$ – użytkownik, dla którego generowane będą rekomendacje
Wyjście: τ_A^* - rekomendacja dopasowana do preferencji użytkownika $\boldsymbol{u_A}$

```
T = []                                                    ▷ utworzenie kolekcji
τ*_A = []                                                 ▷ utworzenie kolekcji
AR := zainicjuj_algorytmy_rekomendacyjne(S)
AR := trenuj_algorytmy_rekomendacyjne(AR, TS)
U_NN := kNN(u_A, U, k)                                    ▷ sąsiedztwo użytkownika u_A

for ar in AR do
    τ = wygeneruj_rekomendacje(u_A, ar)
    T.add(τ)

    for u in U_NN do
        τ = wygeneruj_rekomendacje(u, ar)
        T.add(τ)                                          ▷ rekomendacje sąsiadów
    end for
end for

C := przekształć_do_postaci_macierzy(T)
C := normalizacja(C, "min-max")
w_uA := wyszukaj_wektor_preferencji(C, TS, u_A)           ▷ algorytm DE

for x in I do
    φ(u_A, x) := pobierz_reprezentację_wektorową(C, x)
    f := w_uA^T φ(u_A, x)                                 ▷ funkcja rangująca
    τ*_A.add(x, f)
end for

τ*_A = sortuj_rosnąco(τ*_A)
τ*_A = wybierz_topN_przedmiotów(τ*_A, 10)
return τ*
```

7. Badania eksperymentalne

Eksperymenty zostały przeprowadzone na zbiorze danych *MovieLens 100k*. W tabeli 3 znajdują się podstawowe informacje dotyczące tego zbioru. Rekomendacje generowane były na podstawie sześciu algorytmów rekomendacyjnych. Każdy z algorytmów rekomendacyjnych generował rekomendacje w postaci dziesięciu przedmiotów, które posortowano zgodnie z poziomem ich istotności. Dodatkowo parametry algorytmów rekomendacyjnych zostały wcześniej odpowiednio dostrojone.

Tabela 3: Podstawowe statystyki zbioru danych *MovieLens 100k*

Nazwa statystyki	Wartość
Liczba wystawionych ocen	100 000
Liczba użytkowników	943
Liczba filmów	1682
Możliwe wartości ocen	$\{1, 2, 3, 4, 5\}$
Średnia ocen	3,53
Mediana ocen	4,0
Rzadkość danych	93,695 %

Ze względu na to, że zaproponowany algorytm bazuje na idei agregacji, zostanie on porównany z pięcioma innymi metodami agregującymi, które omówiono w rozdziale 4. Do ewaluacji zaproponowanego podejścia wykorzystane zostaną dedykowane miary, które służą do określenia jakości rankingu. Miary te porównują rekomendacje, które zostały wygenerowane przez poszczególne algorytmy rekomendacyjne z przedmiotami, które znajdują się w zbiorze testowym danego użytkownika. Zostały one szczegółowo opisane w rozdziale 2 i będą obliczane dla następujących wartości odcięcia @ (ang. cut-off treshold): *AP@10*, *NDCG@10*, *NDCG@5*, *Precision@1* oraz *Precision@10*.

Aby potwierdzić skuteczność zaproponowanego podejścia, badania zostały zrealizowane na pełnym zbiorze użytkowników, którzy byli dostępni w zbiorze U, czyli 943 użytkownikach. Dodatkowo w celu wykazania istotności statystycznej prezentowanych wyników, przeprowadzony zostanie test statystyczny Wilcoxona. Symbolem ⇧ będziemy oznaczać, że dany algorytm jest statystycznie lepszy od wszystkich algorytmów rekomendacyjnych. Natomiast ⇧ będziemy oznaczać, że dany algorytm jest statystycznie lepszy od wszystkich metod agregujących.

Dodatkowo w tabeli 4 zaprezentowane zostały wykorzystane w eksperymentach algorytmy rekomendacyjne.

Tabela 4: Algorytmy rekomendacyjne wykorzystane w eksperymentach

Algorytm	Bazujący na	Referencja
UserUser	Sąsiedztwie	[13]
ItemItem	Sąsiedztwie	[11]
PureSVD	Faktoryzacji macierzy	[16]
ImplicitMF	Faktoryzacji macierzy	[17]
BPR	Faktoryzacji macierzy	[21]
MostPopular	Najpopularniejszych przedmiotach	[13]

Tabela 5: Wyniki eksperymentów przedstawiających jakość rekomendacji, która została wygenerowana przez poszczególne algorytmy rekomendacyjne, metody agregujące oraz zaproponowany algorytm. Najlepsze wyniki dla danej miary zostały pogrubione

Algorytm	**MAP@10**	**NDCG@10**	**NDCG@5**	**Precision@1**	**Precision@10**
Algorytmy rekomendacyjne					
BPR	0,1025	0,1796	0,1720	0,2354	0,1671
ImplicitMF	0,1110	0,1879	0,1764	0,2322	0,1754
ItemItem	0,1123	0,1936	0,1878	0,2566	0,1726
SVD	0,1023	0,1783	0,1763	0,2365	0,1604
UserUser	0,1156	0,1946	0,1901	0,2503	0,1764
MostPopular	0,0583	0,1097	0,1013	0,1432	0,1102
Metody agregujące					
BordaFuse	0,1200	0,2057	0,1967	0,2418	0,1888
CombMax	0,0927	0,1633	0,1621	0,2174	0,1499
CombMed	0,1207	0,2025	0,1948	0,2545	0,1857
CombMin	0,1043	0,1833	0,1748	0,2238	0,1680
CombSum	0,1197	0,2040	0,1950	0,2545	0,1874
Zaproponowany algorytm					
EAR_{u_A}	0,1196	0,2042	0,1971	0,2556	0,1867
$EAR_{\_5\_NN}$	0,1292 ⇑ ⇑	0,2152	**0,2086**	**0,2609**	0,1973
$EAR_{\_10\_NN}$	**0,1299** ⇑	**0,2156**	0,2065	0,2545	**0,1985**
$EAR_{\_15\_NN}$	0,1283 ⇑	0,2130	0,2059	0,2577	0,1964

Analizując wyniki przedstawione w tabeli 5 można zauważyć, że algorytmem rekomendacyjnym, który generował rekomendacje najwyższej jakości był algorytm *UserUser*. Natomiast algorytmem, który generował rekomendacje najniższej jakości, był niespersonalizowany algorytm *MostPopular*. Generalnie wszystkie spersonalizowane algorytmy rekomendacyjne, generowały rekomendacje zbliżonej jakości. Ponadto analizując wyniki uzyskane przez metody agregujące, można zauważyć, że agregacja najwyższej jakości została utworzona przy wykorzystaniu metod: *CombMed*, *BordaFuse* oraz *CombSum*. Metodami, które utworzyły agregację

wyraźnie gorszą, były metody: *CombMax* oraz *CombMin*.

W tabeli 5 przedstawione zostały również wyniki, które zostały uzyskane przez zaproponowany algorytm *EAR*. Gdy agregacja była tworzona tylko na podstawie rekomendacji wygenerowanych dla aktywnego użytkownika (oznaczenie EAR_{u_A}), algorytm ten uzyskał wyniki, które były lepsze od wszystkich algorytmów rekomendacyjnych. Niestety był on gorszy od algorytmów agregujących *BordaFuse*, *CombMed* oraz *CombSum* (dla miary $AP@10$). Analizując jednak wyniki, jakie zostały uzyskane przez algorytm *EAR* z modyfikacją uwzględniającą najbliższe sąsiedztwo (oznaczonych odpowiednio: $\text{EAR}_{\text{\_5\_NN}}$, $\text{EAR}_{\text{\_10\_NN}}$, $\text{EAR}_{\text{\_15\_NN}}$), można zauważyć, że jakość rekomendacji utworzonej na podstawie takiej agregacji wzrosła. Świadczy to o tym, że uwzględnienie rankingów innych użytkowników w procesie agregacji, może pozytywnie wpłynąć na jakość generowanej rekomendacji.

Należy jednak zauważyć, że liczba użytkowników (sąsiadów), którzy są uwzględniani w procesie agregacji również ma znaczenie. Analizując wyniki przedstawione w tabeli 5 można zauważyć, że uwzględniając rankingi pięciu najbliższych sąsiadów użytkownika u_A, algorytm ten uzyskał najlepsze wyniki dla miar *NDCG@5* oraz *Precision@1*. Świadczy to o tym, że taka liczba podobnych użytkowników wpływała lepiej na rekomendacje, które były prezentowane na początkowych pozycjach w rankingu. Natomiast uwzględnienie w procesie agregacji rankingów dziesięciu najbliższych sąsiadów, zwiększało jakość rekomendacji, która była wyrażona przy pomocy miar: *AP@10*, *NDCG@10* oraz *Precision@10*. Świadczy to o tym, że algorytm ten uzyskiwał lepsze wyniki dla pierwszych dziesięciu przedmiotów, które były prezentowane użytkownikowi.

Należy jednak zauważyć, że zwiększanie liczby sąsiadów niekoniecznie musi wiązać się ze zwiększeniem jakości generowanych rekomendacji. Przykładowo, przy uwzględnieniu rankingów wygenerowanych dla piętnastu najbliższych sąsiadów aktywnego użytkownika u_A, jakość generowanej rekomendacji nie zwiększyła się. Świadczy to o tym, że liczba użytkowników, którzy są uwzględniani w procesie agregacji nie może być zbyt duża, ponieważ zbyt oddaleni użytkownicy wprowadzają za dużo szumu informacyjnego do tworzonej agregacji, co w następstwie powoduje spadek jakości rekomendacji.

8. Podsumowanie

W rozprawie zaproponowano algorytm agregujący, który wykorzystując historyczne interakcje użytkownika z systemem, modeluje jego preferencje. Preferencje te reprezentowane są przez wektor liczb rzeczywistych, gdzie poszczególne elementy tego wektora określają w jakim stopniu rekomendacja wygenerowana przez dany algorytm, będzie wpływać na końcową agregację. Do wyszukania tego wektora wykorzystany został algorytm ewolucji różnicowej, który umożliwia bezpośrednią optymalizację funkcji rangującej w stosunku do miar, służących do oceny jakości wygenerowanych rekomendacji. Takie podejście umożliwiło stworzenie nadzorowanego algorytmu agregującego, którego efektywność została zweryfikowana przez liczne eksperymenty przeprowadzone na popularnym i ogólnodostępnym zbiorze danych *MovieLens 100k*, a wyniki badań zostały potwierdzone przez testy statystyczne.

Na bazie wyników eksperymentów zaprezentowanych w rozprawie, wyciągnięte zostały trzy główne wnioski:

1. Badania przy wykorzystaniu klasycznych algorytmów agregujących wykazały, że ich zastosowanie poprawia jakość generowanych rekomendacji. Z tego względu zastosowanie metod agregujących w systemach rekomendacji jest uzasadnione. Metodami, które uzyskały najlepsze wyniki były: *BordaFuse*, *CombMed*, *CombSum*.

2. Badania przeprowadzone dla zaproponowanego algorytmu *EAR* wykazały, że z powodzeniem można wykorzystać algorytm DE do bezpośredniej optymalizacji miary AP, w celu uzyskania wektora preferencji aktywnego użytkownika u_A. Algorytm ten, dzięki zaproponowanym wariantom funkcji oceny, przeważnie uzyskiwał wyniki lepsze od algorytmów rekomendacyjnych oraz klasycznych metod agregujących.

3. Zasadniczą zaletą zaproponowanego podejścia jest łatwość jego wdrożenia do istniejącego systemu rekomendacyjnego. Ze względu na to, że agregacja odbywa się tylko na podstawie wygenerowanych rankingów, nie występuje tutaj potrzeba ingerencji w implementacje samych algorytmów rekomendacyjnych.

Publikacje autora

1. M. **Bałchanowski**, U. Boryczka. A Comparative Study of Rank Aggregation Methods in Recommendation Systems. *Entropy*, wolumen 25(1), 2023. (Punktacja MNiSW: **100**)

2. M. **Bałchanowski**, U. Boryczka. Aggregation of Rankings Using Metaheuristics in Recommendation Systems. *Electronics*, wolumen 11(3), 2022. (Punktacja MNiSW: **100**)

3. M. **Bałchanowski**, U. Boryczka. Collaborative Rank Aggregation in Recommendation Systems. *Procedia Computer Science*, wolumen 207, 2022, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES 2022, strony 2213–2222. (Punktacja MNiSW: **70**)

4. U. Boryczka, M. **Bałchanowski**. Speed up Differential Evolution for ranking of items in recommendation systems. *Procedia Computer Science*, wolumen 192, 2021, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES 2021, strony 2229–2238. (Punktacja MNiSW: **70**)

5. U. Boryczka, M. **Bałchanowski**. Using Differential Evolution in order to create a personalized list of recommended items. *Procedia Computer Science*, wolumen 176, 2020, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES 2020, strony 1940–1949. (Punktacja MNiSW: **70**)

6. M. **Bałchanowski**, U. Boryczka, K. Dworak. Using Differential Evolution with a Simple Hybrid Feature for Personalized Recommendation. *Intelligent Information and Database Systems. ACIIDS 2018.* Red. N. T. Nguyen et al., Cham: Springer International Publishing, 2018, strony 137–146. (Punktacja MNiSW: **20**)

7. U. Boryczka, M. **Bałchanowski**. Differential Evolution in a Recommendation System Based on Collaborative Filtering. *Computational Collective Intelligence. ICCCI 2016.* Red. N. T. Nguyen et al., Cham: Springer International Publishing, 2016, strony 113–122. (Punktacja MNiSW: **20**)

Bibliografia

[1] C. C. Aggarwal. Advanced Topics in Recommender Systems. *Recommender Systems: The Textbook*, Cham: Springer International Publishing, 2016, strony 411–448.

[2] V. W. Anelli, A. Bellogín, T. Di Noia, D. Jannach, C. Pomo. Top-N Recommendation Algorithms: A Quest for the State-of-the-Art. *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '22, Barcelona, Spain: Association for Computing Machinery, 2022, strony 121–131.

[3] D. Bollegala, N. Noman, H. Iba. RankDE: Learning a ranking function for information retrieval using differential evolution. *Genetic and Evolutionary Computation Conference, GECCO'11*, 2011, strony 1771–1778.

[4] J.-C. de Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[8] A. Brown, W. Xie, V. Kalogeiton, A. Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. *European Conference on Computer Vision*, Springer, 2020, strony 677–694.

[9] D. J. Brown. Aggregation of Preferences. *The Quarterly Journal of Economics*, wolumen 89(3), 1975, strony 456–469.

[10] P. Cichosz. Uczenie się aproksymacji funkcji. *Systemy uczące się*, Wydawnictwa Naukowo-Techniczne, 2000, strony 432–492.

[11] M. Deshpande, G. Karypis. Item-Based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.*, wolumen 22(1), 2004, strony 143–177.

[12] C. Dwork, R. Kumar, M. Naor, D. Sivakumar. Rank Aggregation Methods for the Web. *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, Hong Kong, Hong Kong: Association for Computing Machinery, 2001, strony 613–622.

[13] M. D. Ekstrand. LensKit for Python: Next-Generation Software for Recommender Systems Experiments. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, Virtual Event, Ireland: Association for Computing Machinery, 2020, strony 2999–3006.

[14] K. Falk. Rangowanie i nauka rangowania. *Praktyczne systemy rekomendacji*, Wydawnictwo Naukowe PWN, 2020, strony 385–414.

[15] E. A. Fox, J. A. Shaw. Combination of Multiple Searches. *TREC*, wolumen 500-215, NIST Special Publication, National Institute of Standards i Technology (NIST), 1993, strony 243–252.

[16] N. Halko, P. G. Martinsson, J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, wolumen 53(2), 2011, strony 217–288.

[17] Y. Hu, Y. Koren, C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining*, 2008, strony 263–272.

[18] A. Karatzoglou, L. Baltrunas, Y. Shi. Learning to Rank for Recommender Systems. *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, Hong Kong, China: Association for Computing Machinery, 2013, strony 493–494.

[19] T. Morzy. Kombinacja klasyfikatorów. *Eksploracja danych. Metody i algorytmy*, Wydawnictwo Naukowe PWN, 2013, strony 313–345.

[20] S. E. L. Oliveira, V. Diniz, A. Lacerda, L. Merschmanm, G. L. Pappa. Is Rank Aggregation Effective in Recommender Systems? An Experimental Analysis. *ACM Trans. Intell. Syst. Technol.*, wolumen 11(2), 2020.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, Montreal, Quebec, Canada: AUAI Press, 2009, strony 452–461.

[22] R. Storn, K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization*, wolumen 11(4), 1997, strony 341–359.

Tabela 6: Spis najważniejszych symboli, które występują w autoreferacie

Symbol	Opis
U	zbiór wszystkich użytkowników
I	zbiór wszystkich przedmiotów
R	macierz preferencji (ocen) użytkownik-przedmiot o rozmiarze $\|U\| \times \|I\|$
S	zbiór n algorytmów rekomendacyjnych $S = \{S^1,\ S^2,\ ...,\ S^n\}$
S^r	algorytm r znajdujący się w zbiorze S
u	ogólny użytkownik systemu rekomendacyjnego
u_i	konkretny użytkownik systemu rekomendacyjnego
u_A	aktywny użytkownik systemu rekomendacyjnego, czyli taki, dla którego generowane są rekomendacje
x	ogólny *przedmiot* znajdujący się w systemie rekomendacyjnym
x_j	konkretny *przedmiot* znajdujący się w systemie rekomendacyjnym
τ	ogólny ranking
τ_A^r	ranking zarekomendowany użytkownikowi u_A przez algorytm S^r
$\tau\left(x_j\right)$	pozycja przedmiotu x_j w rankingu τ
Ψ	funkcja agregująca
τ_A^*	ranking utworzony po procesie agregacji dla użytkownika u_A
$\mathcal{T}$	zbiór n rankingów $\mathcal{T} = \{\tau^1, \tau^2, \ldots, \tau^n\}$
$\mathcal{T}_A$	zbiór rankingów, które zostały zarekomendowane aktywnemu użytkownikowi u_A, przez algorytmy w zbiorze S
N_A	zbiór użytkowników, wyznaczonych jako sąsiedztwo aktywnego użytkownika u_A, przy pomocy algorytmu kNN
$\mathcal{T}_{N_A}$	zbiór rankingów, które zostały zarekomendowane przez algorytmy w zbiorze S, użytkownikom w zbiorze N_A
w	wektor, który jest również osobnikiem w algorytmie DE
w_{u_A}	wektor preferencji aktywnego użytkownika u_A
P	populacja algorytmu DE
$\phi(\mathcal{T}_A, \mathcal{T}_{N_A}, I)$	jest to reprezentacja wektorowa przedmiotów $x_j \in I$, która powstała na bazie wartości, które przedmioty te mają przyporządkowane w rankingach znajdujących się w zbiorach $\mathcal{T}_A$ oraz $\mathcal{T}_{N_A}$
$\phi(U)$	jest to reprezentacja wektorowa wszystkich użytkowników w zbiorze U