POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Marcin Połajdowicz
Nr albumu: 184265
Poziom kształcenia: studia drugiego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Specjalność: Algorytmy i technologie internetowe

Imię i nazwisko studenta: Maciej Sztramski
Nr albumu: 184779
Poziom kształcenia: studia drugiego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Specjalność: Aplikacje rozproszone i systemy internetowe

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Modelowanie optymalnych sposobów zakupu licencji oprogramowania w sieciach społecznościowych za pomocą dominowania w grafach

Tytuł pracy w języku angielskim: Modeling optimal ways to purchase software licenses in social networks via graph domination

Opiekun pracy: dr inż. Joanna Raczek

Streszczenie

Celem niniejszej pracy magisterskiej było opracowanie grafowego modelu optymalizacji zakupu licencji oprogramowania (np. Duolingo Super, Spotify Premium) w sieciach społecznościowych. Problem ten sformułowano jako rozszerzenie klasycznego dominowania (w tym dominowania rzymskiego) w grafach, przy czym uwzględniono praktyczne ograniczenia dotyczące rozmiaru grup licencyjnych oraz struktury kosztów licencji indywidualnych i grupowych.

W pracy przeprowadzono analizę złożoności obliczeniowej problemu, wykazując jego NP-trudność. Zaimplementowano i porównano podejścia algorytmiczne: metody dokładne (ILP/MIP) oraz kilka metod przybliżonych i metaheurystyk (algorytm zachłanny, algorytm genetyczny, tabu search, symulowane wyżarzanie, ant colony optimization). Badania eksperymentalne wykonano na rzeczywistych ego-sieciach Facebooka oraz na danych syntetycznych (Barabási-Albert, Erdős-Rényi, Watts-Strogatz).

Wyniki pokazują, że metaheurystyki (w szczególności algorytm genetyczny, tabu search i symulowane wyżarzanie) pozwalają osiągać rozwiązania bliskie optymalnym przy znacznym skróceniu czasu obliczeń względem ILP na większych instancjach. Dodatkowo przeanalizowano wariant dynamiczny, w którym sieć ulega zmianom w czasie, oraz wpływ polityk cenowych i dodatkowych typów planów subskrypcyjnych na optymalne decyzje.

Słowa kluczowe: teoria grafów, optymalizacja kombinatoryczna, dominowanie rzymskie, sieci społecznościowe, heurystyki, metaheurystyki, programowanie całkowitoliczbowe.

Abstract

This thesis develops a graph-based model for optimizing software license purchases (e.g., Duolingo Super, Spotify Premium) in social networks. The problem is formulated as an extension of classical domination (including Roman domination) in graphs, incorporating realistic constraints on group sizes and pricing schemes for individual and group licenses.

We analyze the computational complexity and show NP-hardness. We implement and compare exact methods (integer linear programming) with heuristic and metaheuristic approaches (greedy, genetic algorithm, tabu search, simulated annealing, ant colony optimization). Experiments are conducted on real-world Facebook ego networks and on synthetic graphs (Barabási-Albert, Erdős-Rényi, Watts-Strogatz).

Results indicate that metaheuristics (notably genetic algorithm, tabu search and simulated annealing) achieve near-optimal solutions with significantly lower runtime than ILP for larger instances. We also study a dynamic variant in which the network evolves over time and evaluate the impact of pricing policies and additional subscription types on optimal purchasing decisions.

Keywords: graph theory, combinatorial optimization, Roman domination, social networks, heuristics, metaheuristics, integer programming.

SPIS TREŚCI

SPIS RYSUNKÓW

SPIS TABEL

LIST OF ALGORITHMS

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

Oznaczenia

$G = (V, E)$	–	graf relacji społecznych.
V, E	–	zbiory wierzchołków i krawędzi.
$N(v),\ N[v]$	–	otoczenie wierzchołka v oraz otoczenie domknięte.
$\deg(v)$	–	stopień wierzchołka v.
$n = \|V\|,\ m = \|E\|$	–	liczba wierzchołków i krawędzi.
Δ	–	maksymalny stopień grafu.
$f : V \to \{0, 1, 2\}$	–	etykietowanie ról (0-odbiorca, 1-indywidualna, 2-grupowa).
$\mathrm{cost}(f)$	–	koszt rozwiązania (def. w rozdz. 2.2).
$\mathcal{L}$	–	rodzina typów licencji $\ell_t = (c_t,\, \ell_t^{\min},\, \ell_t^{\max})$, w skrócie $(c_t,\, l_t,\, u_t)$.
I, G, R	–	zbiory: licencji indywidualnych, grupowych i odbiorców.
p	–	względny koszt licencji grupowej $p = c_g/c_1$.

Skróty

SaaS	–	Software as a Service.
ILP/MIP	–	(Mixed) Integer Linear Programming.
GA	–	Genetic Algorithm.
SA	–	Simulated Annealing.
TS	–	Tabu Search.
ACO	–	Ant Colony Optimization.
MDS	–	Minimum Dominating Set.
RD	–	Roman Domination.

1. WPROWADZENIE

1.1 Wstęp i motywacja

W ostatnich latach coraz większe znaczenie zyskują modele subskrypcyjne w sektorze oprogramowania i usług cyfrowych. Zgodnie z indeksem gospodarki subskrypcyjnej rynek ten zwiększył swoją wartość o ponad 400% od roku 2012 do roku 2021 [1], a w 2024 roku osiągnął przybliżoną wartość blisko 600 miliardów dolarów [2].

Konsumenci coraz częściej opłacają regularne abonamenty zamiast jednorazowych zakupów, co zapewnia firmom stałe przychody, a użytkownikom wygodny dostęp do usług. Wiele popularnych platform, w tym aplikacje edukacyjne, serwisy streamingowe czy oprogramowanie w modelu Software as a Service (SaaS), opiera się na modelu subskrypcyjnym. Serwisy streamingowe, takie jak Spotify czy Netflix, często oferują plany rodzinne, w których kilka osób może współdzielić jedną subskrypcję grupową, której koszt jest niższy niż zakup kilku subskrypcji indywidualnych. W podobny sposób platforma edukacyjna do nauki języków Duolingo udostępnia plan rodzinny Super Duolingo (ang. Duolingo Super Family) [3], który pozwala grupie znajomych lub osób spokrewnionych współdzielić korzyści subskrypcji premium. Zachęca to użytkowników do grupowego zakupu subskrypcji, redukując jednocześnie koszt przypadający na jedną osobę. Subskrypcja wiąże się z czasowym nabyciem licencji, dlatego w dalszej części pracy oba pojęcia będą używane zamiennie.

W przypadku formowania się takich grup kontekst społeczny odgrywa istotną rolę. Rozsądne i optymalne korzystanie z subskrypcji grupowych wymaga, aby główny posiadacz subskrypcji znał osoby zainteresowane wspólnym korzystaniem z usługi, czy to ze względu na chęć podziału kosztów, podobne zainteresowania lub zwyczajną bliskość relacji. Użytkownicy często łączą się w grupy z rodziną czy przyjaciółmi, ponieważ łatwiej wtedy o zaufanie w zakresie współdzielenia konta. Innym powodem może być też wygoda, ponieważ jedna wspólna subskrypcja upraszcza zarządzanie płatnościami i zapewnia wszystkim członkom grupy taki sam dostęp do usługi. Takie czynniki społeczne mają więc istotny wpływ na to, jak faktycznie kształtują się struktury współdzielenia w sieciach użytkowników. Rozwój mediów społecznościowych i komunikatorów ułatwia zawieranie takich porozumień w gronie znajomych lub osób o podobnych zainteresowaniach. W praktyce często dochodzi do sytuacji, w których użytkownicy umawiają się na wspólny zakup abonamentu. Sieć powiązań społecznych determinuje, kto z kim może skutecznie współdzielić licencję.

Analiza przedstawionych mechanizmów prowadzi do sformułowania problemu optymalizacyjnego: jak zaplanować zakup licencji w grupie powiązanych użytkowników, aby zminimalizować łączny koszt dostępu do usługi. Innymi słowy, mając daną sieć znajomości oraz dostępne opcje licencyjne, należy dobrać podzbiór użytkowników kupujących licencje (oraz rodzaj tych licencji)

tak, by wszyscy użytkownicy mieli dostęp do usługi przy możliwie najniższym koszcie. Intuicyjnie jest to problem pokrycia grafu zbiorem *właścicieli* (osób wykupujących licencje), w taki sposób, by każdy wierzchołek był albo objęty licencją, albo sąsiadował z kimś, kto licencję posiada.

Warto zauważyć, że opisana struktura problemu ma ścisłe powiązania z zagadnieniami teorii grafów. Jest ona bliska klasycznemu problemowi dominowania, ponieważ wybór użytkowników kupujących licencje grupowe pełni tę samą funkcję co wybór zbioru dominującego. W obu przypadkach chodzi o to, aby wybrany zestaw wierzchołków pokrywał całą sieć. W dalszej części pracy został również pokazany związek z wariantem znanym jako *dominowanie rzymskie*. Takie odniesienia do teorii grafów stanowią istotne uzupełnienie głównego celu badań, którym jest optymalizacja kosztów licencji w społeczności użytkowników.

1.2 Cele i zakres pracy

Celem niniejszej pracy jest formalizacja i analiza problemu optymalnego zakupu licencji w sieciach społecznościowych, zaproponowanie metod jego rozwiązania oraz weryfikacja przyjętych rozwiązań. W pierwszej kolejności opracowany został model grafowy opisujący powiązania między użytkownikami oraz różne strategie zakupowe wraz z odpowiadającymi im kosztami. Taki model pozwolił zdefiniować problem minimalizacji kosztów jako zadanie optymalizacyjne na grafie. Następnie wykazano ścisły związek z problemem dominowania w grafach. W szczególności pokazano, że dla pewnej klasy modeli licencjonowania zadanie optymalnego doboru subskrypcji jest równoważne znalezieniu minimalnego zbioru dominującego lub rozwiązaniu pokrewnego problemu *dominowania rzymskiego*. Odwołanie do znanych wyników o dominowaniu obejmuje zarówno aspekty złożoności obliczeniowej, jak i badania nad algorytmami aproksymacyjnymi, co pozwala lepiej zrozumieć trudności badanego problemu oraz zaprojektować efektywne metody jego rozwiązywania.

Zakres pracy obejmuje analizę teoretyczną badanego problemu oraz metody algorytmiczne jego rozwiązania, uzupełnione o eksperymenty obliczeniowe. Rozpatrzone zostały różne modele cenowe licencji, zarówno hipotetyczne, jak i rzeczywiste. Do pierwszej grupy należą warianty nawiązujące do dominowania rzymskiego, w których koszt licencji grupowej stanowi wielokrotność ceny licencji indywidualnej. W drugiej grupie znajdują się modele oparte na rzeczywistych ofertach usług, takich jak Spotify, Netflix czy Duolingo, co pozwala zweryfikować otrzymane wyniki w kontekście praktycznych scenariuszy.

Osobno przeanalizowane zostały warianty, w których decyzje zakupowe podejmowane są globalnie (jednocześnie dla całej społeczności), oraz scenariusze dynamiczne. W wersji dynamicznej zakupy realizowane są w kolejnych krokach czasowych, a struktura sieci społecznościowej może się zmieniać (rozszerzanie lub zmniejszanie liczby użytkowników, powstawanie lub zanik relacji). Ze względu na wysoką złożoność obliczeniową problemu przedstawione zostały zarówno metody dokładne, jak i heurystyczne. Metody dokładne gwarantują znalezienie rozwiązania optymalnego, lecz ich czas działania szybko rośnie wraz z rozmiarem grafu. Z kolei metody heurystyczne nie dają gwarancji optymalności, ale pozwalają uzyskać rozwiązania dobrej jakości

w akceptowalnym czasie. Celem praktycznym jest wskazanie podejść skutecznych w optymalizacji kosztów subskrypcji w dużych sieciach społecznościowych oraz identyfikacja czynników najsilniej wpływających na wyniki, co zilustrowane jest wynikami eksperymentów.

Podsumowując, w ramach pracy zrealizowane zostały następujące zadania badawcze i implementacyjne:

- Sformalizowano model optymalizacji zakupu licencji w sieciach społecznościowych, obejmujący etykietowanie ról $f : V \to \{0, 1, 2\}$, warunki wykonalności (pokrycie, sąsiedztwo, pojemność) oraz funkcję kosztu.
- Wykazano powiązania z *dominowaniem rzymskim* i sformułowano twierdzenie o równoważności w szczególnym przypadku kosztów i pojemności; omówiono również konsekwencje złożonościowe i aproksymacyjne.
- Zaimplementowano i porównano różne metody: ILP (PuLP/CBC), algorytm zachłanny, metaheurystyki (algorytm genetyczny, symulowane wyżarzanie, przeszukiwanie tabu, algorytm mrówkowy).
- Opracowano środowisko eksperymentalne dla grafów syntetycznych i ego-sieci Facebooka, wraz z pomiarem czasu działania i kosztu.
- Przeanalizowano wariant dynamiczny (mutacje grafu) oraz porównano podejścia cold-start i warm-start dla metaheurystyk.
- Zbadano wpływ polityk cenowych i typów planów (Individual/Duo/Family) na strukturę rozwiązań i koszt całkowity.

1.3 Struktura

Struktura pracy została zorganizowana w dziewięciu rozdziałach. Obejmują one część wprowadzającą, w której przedstawiono tło i motywację podjętego zagadnienia, część analityczno-badawczą, zawierającą opis zaproponowanych modeli oraz przeprowadzonych eksperymentów, a także część podsumowującą, w której sformułowano wnioski oraz wskazano możliwe kierunki dalszych badań. Poniżej zaprezentowano opis treści wszystkich rozdziałów, stanowiących kolejne etapy realizacji pracy.

Rozdział 1 – Wprowadzenie. Przedstawia tło problemu, motywację podjęcia tematu oraz znaczenie optymalizacji kosztów w kontekście współdzielenia licencji w sieciach społecznościowych. Określone są cele i zakres pracy oraz jej ogólna struktura.

Rozdział 2 – Model grafowy i analiza problemu. Definiuje reprezentację sieci społecznościowej w postaci grafu oraz formalny opis problemu optymalizacji kosztów. Uwzględnione są przyjęte założenia, definicje pojęć oraz warianty wynikające z odmiennych modeli cenowych i ograniczeń pojemnościowych. Rozdział stanowi podstawę do dalszych rozważań algorytmicznych.

Rozdział 3 – Związek z dominowaniem w grafach. Omawia pojęcie zbiorów dominujących i dominowania rzymskiego, pokazując, w jaki sposób badany problem można interpretować w tych

kategoriach. Przedstawione są również aspekty złożoności obliczeniowej, w tym dowód NP-trudności, oraz wynikające z tego konsekwencje dla możliwości projektowania algorytmów.

Rozdział 4 – Dane testowe. Opisuje rodzaje grafów wykorzystanych w eksperymentach: syntetyczne, generowane przy pomocy popularnych modeli (Barabási–Albert, Watts–Strogatz oraz Erdős–Rényi), oraz grafy rzeczywiste pochodzące z repozytoriów badawczych. Uwzględniono także sposób przygotowania instancji testowych oraz narzędzia użyte do wizualizacji sieci.

Rozdział 5 – Metody algorytmiczne optymalizacji kosztów licencji. Rozdział skupia się na części obliczeniowej. Przedstawiona jest formalizacja problemu w postaci programu całkowitoliczbowego oraz opis metod dokładnych, które mogą znaleźć optymalne rozwiązania dla mniejszych instancji. Omówione są także podejścia przybliżone i heurystyczne.

Rozdział 6 – Eksperymenty i analiza wyników. Przedstawia proces oceny algorytmów na przygotowanych danych testowych. Określone są kryteria porównawcze (m.in. czas działania, złożoność obliczeniowa, uzyskane koszty), a następnie zaprezentowane wyniki eksperymentów dla różnych typów grafów i ich skal. Analizowany jest wpływ parametrów algorytmów na efektywność i jakość uzyskiwanych rozwiązań.

Rozdział 7 – Analiza dynamicznej wersji problemu. Rozważany jest scenariusz, w którym zakupy licencji odbywają się sekwencyjnie, a struktura sieci może ulegać zmianie w czasie. Opisane są możliwe adaptacje algorytmów do takiej sytuacji oraz przeprowadzone eksperymenty badające ich skuteczność. Poruszona jest także kwestia stabilności i elastyczności strategii w środowisku dynamicznym.

Rozdział 8 – Rozszerzenia modelu. Omawia dodatkowe aspekty, które mogą wpływać na decyzje optymalizacyjne, takie jak polityki cenowe oraz zróżnicowanie typów licencji. Analizowane są również potencjalne ograniczenia modelu i możliwości jego dalszego uogólnienia.

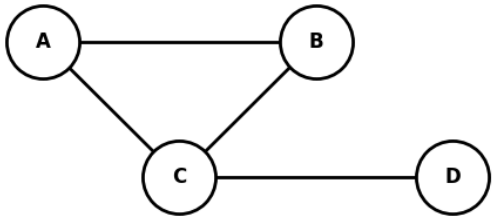
Rozdział 9 – Podsumowanie i wnioski. Zawiera syntetyczne zestawienie wyników pracy. Wskazuje, w jakim stopniu zrealizowane zostały założone cele, oraz proponuje kierunki dalszych badań, w tym rozwój modeli, udoskonalenia algorytmów oraz badania nad skalowalnością i praktycznymi zastosowaniami. Rozdział ten zamyka całość pracy, odpowiadając na pytania badawcze i wskazując, jak uzyskane rezultaty mogą zostać wykorzystane w praktyce.

2. MODEL GRAFOWY PROBLEMU ZAKUPU LICENCJI

2.1 Reprezentacja grafowa

Aby formalnie opisać zjawisko współdzielenia licencji, sieć relacji społecznych modelujemy jako graf nieskierowany $G = (V, E)$. Każdy wierzchołek $v \in V$ reprezentuje pojedynczego użytkownika, natomiast krawędź $\{u, v\} \in E$ oznacza, że użytkownicy u i v znajdują się w relacji umożliwiającej współdzielenie licencji grupowej. Graf jest nieskierowany, ponieważ zakładamy symetryczność tej relacji: jeśli u zna v, to również v zna u.

Przyjmujemy również, że graf G nie zawiera pętli, tj. $\{v, v\} \notin E$ dla każdego $v \in V$, ani krawędzi wielokrotnych - każda para użytkowników może być powiązana co najwyżej jedną krawędzią. Przykład takiej struktury zilustrowano na Rysunku 2.1.



Rysunek 2.1: Przykładowy graf relacji społecznych między użytkownikami.

Opisana reprezentacja, w której wierzchołki odpowiadają jednostkom, a krawędzie bezpośrednim relacjom umożliwiającym interakcję, jest powszechnie stosowana w analizie sieci społecznościowych [4, 5]. Takie ujęcie pozwala formalnie modelować i badać zjawiska zachodzące w społecznościach użytkowników usług cyfrowych.

W przyjętym modelu zakłada się, że współdzielenie licencji może odbywać się wyłącznie między osobami połączonymi bezpośrednią krawędzią w grafie. Licencja grupowa oznacza w tym kontekście typ umożliwiający współdzielenie dostępu przez właściciela oraz jego bezpośrednich sąsiadów. Oznacza to, że użytkownicy muszą znać się bezpośrednio i mieć wzajemne zaufanie, co jest istotne na przykład w przypadku przekazywania danych logowania lub zapraszania do planu rodzinnego. Relacje pośrednie, w których użytkownicy są powiązani poprzez wspólnych znajomych (np. $A \sim B$ oraz $B \sim C$, lecz brak bezpośredniego powiązania $A \sim C$), nie są uwzględniane w analizie. Oznacza to, że dla danego grafu $G = (V, E)$, w którym V to zbiór użytkowników, a $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$ to zbiór relacji znajomości, analizie podlegają wyłącznie relacje bezpośrednie, czyli pary $\{u, v\} \in E$. Na przykład w grafie przedstawionym na Rysunku 2.1, użytkownicy A i D są wprawdzie połączeni za pośrednictwem ścieżki $A \to C \to D$, jednak ponieważ brakuje bezpośredniego połączenia $\{A, D\} \in E$, to taka relacja nie jest uznawana za podstawę do współdzielenia licencji w tym modelu. Mimo że w praktyce relacje pośrednie, takie jak ścieżki

długości większej niż jeden, mogą sprzyjać tworzeniu grup subskrypcyjnych, w analizie zostały one pominięte w celu uproszczenia problemu.

Graf społecznościowy nie musi być pełny. Dopuszcza się dowolną strukturę odpowiadającą rzeczywistym relacjom społecznym. W analizie istotną rolę odgrywa stopień wierzchołków, ponieważ decyduje on o liczbie osób, którym dany użytkownik może udostępnić swoją licencję. Dla wierzchołka $v \in V$, jego stopień oznaczamy przez $\deg(v)$, co odpowiada liczbie sąsiadów użytkownika v w grafie $G = (V, E)$.

Należy jednak zauważyć, że nawet w przypadku wysokiego stopnia $\deg(v)$, użytkownik niekoniecznie może współdzielić licencję ze wszystkimi swoimi sąsiadami. Ograniczenia techniczne, takie jak limity liczby współużytkowników narzucane przez dostawcę usługi, sprawiają, że liczba osób objętych jedną licencją grupową pozostaje ograniczona. W analizowanym modelu wprowadzamy zatem parametr k, oznaczający pojemność licencji grupowej, czyli maksymalną liczbę osób, wliczając w to właściciela, które mogą korzystać z jednej licencji.

2.2 Definicja problemu

Optymalizacja kosztu dostępu do usługi wymaga przypisania wszystkim wierzchołkom grafu $G = (V, E)$ odpowiednich ról. Każdy użytkownik $v \in V$ może uzyskać dostęp do usługi na trzy sposoby:

1. Poprzez wykupienie licencji indywidualnej.
2. Poprzez wykupienie licencji grupowej.
3. Jako odbiorca, korzystający z licencji grupowej należącej do innego użytkownika.

Licencja indywidualna zapewnia dostęp wyłącznie jej właścicielowi, natomiast licencja grupowa umożliwia współdzielenie dostępu z maksymalnie $k - 1$ sąsiadami w grafie.

Dla przejrzystego i precyzyjnego zdefiniowania modelu wprowadzamy trzy zbiory reprezentujące użytkowników, czyli węzły posiadające własną licencję bądź korzystające z licencji innego węzła:

- I - posiadacze licencji indywidualnych;
- G - posiadacze licencji grupowych;
- R - odbiorcy, którzy sami nie kupują licencji, lecz korzystają z licencji innego użytkownika.

Aby formalnie opisać ten podział, wprowadzamy etykietowanie ról $f : V \to \{0, 1, 2\}$. Wartość 0 oznacza węzeł będący odbiorcą, 1 odpowiada licencji indywidualnej, a 2 licencji grupowej. Takie przypisanie nawiązuje do klasycznego problemu dominowania rzymskiego, w którym wierzchołki również otrzymują etykiety z tego samego zbioru wartości. Odpowiadające zbiory definiujemy jako $I = \{v : f(v) = 1\}$, $G = \{v : f(v) = 2\}$ oraz $R = V \setminus (I \cup G)$. Przyjmujemy zbiór dostępnych typów licencji

$$\mathcal{L} = \{\ell_t = (c_t, m_t, k_t) \mid t = 1, 2, \ldots, T\}, \tag{2.1}$$

gdzie:

- c_t - koszt licencji typu t,
- m_t – minimalna liczba użytkowników wliczając właściciela,
- k_t – maksymalna liczba użytkowników wliczając właściciela,
- T – całkowita liczba dostępnych typów licencji.

W podstawowym wariancie rozważanym w tym rozdziale zakładamy, że zbiór $\mathcal{L}$ opisany we wzorze (2.1) obejmuje wyłącznie dwa typy licencji: indywidualną oraz jedną grupową, a zatem przyjęte jest $T = 2$. Model ten nie przewiduje sytuacji z wieloma rodzajami planów grupowych (np. Duo, Family), lecz ogranicza się do najprostszego przypadku odpowiadającego rozwiązaniom takim jak w Duolingo.

Warunki wykonalności. Spełnienie rozwiązania wymaga, aby dla etykietowania $f : V \to \{0, 1, 2\}$ zachodziły następujące warunki:

1. Pokrycie: Każdy użytkownik $v \in V$ musi mieć dostęp do usługi, tj. $f(v) \in \{1, 2\}$ lub istnieje sąsiad $u \in N(v)$ taki, że $f(u) = 2$ i v jest przypisany do grupy u.
2. Sąsiedztwo: Odbiorca $v \in R$ może być przypisany tylko do właściciela $u \in G$ z $\{u, v\} \in E$.
3. Pojemność: Liczba użytkowników przypisanych do właściciela $u \in G$ (wraz z nim samym) musi spełniać $m \leq 1 + |R_u| \leq k$, gdzie R_u oznacza zbiór odbiorców korzystających z licencji u.

Funkcja kosztu. W wariancie podstawowym, w którym $T = 2$, całkowity koszt rozwiązania wyraża się zależnością:

$$\text{cost}(f) = |I| \cdot c_i + |G| \cdot c_g, \tag{2.2}$$

gdzie:

- I - zbiór użytkowników z licencją indywidualną,
- G - zbiór użytkowników z licencją grupową,
- c_i - koszt licencji indywidualnej,
- c_g - koszt licencji grupowej.

Cel optymalizacji. Celem problemu jest minimalizacja funkcji kosztu (2.2) dla danego grafu $G = (V, E)$, przy spełnieniu wszystkich istotnych warunków wykonalności.

2.3 Koszty i ograniczenia

2.3.1 Ograniczenia techniczne i społeczne współdzielenia licencji

Kluczowymi parametrami modelu są minimalna oraz maksymalna liczba osób, które mogą współdzielić jedną licencję grupową. Maksymalny rozmiar grupy oznaczamy przez k i wliczamy do niego także użytkownika nabywającego licencję. Parametr k jest zwykle narzucany przez dostawcę usługi. Przykładowo, plan rodzinny Spotify Premium pozwala na korzystanie maksymalnie sześciu osobom (właściciel + pięć członków rodziny), co odpowiada wartości $k = 6$.

Analogicznie wprowadzamy parametr m, który określa minimalną liczbę osób niezbędnych do utworzenia grupy. Oznacza to, że licencja grupowa jest ważna tylko wtedy, gdy zostanie wykorzystana przez co najmniej m osób (łącznie z właścicielem). Przykładem jest tzw. plan „Duo", w którym licencję mogą współdzielić dokładnie dwie osoby ($m = 2, k = 2$). W innych przypadkach m może przyjmować wartości mniejsze niż k, np. $m = 2, k = 6$ dla planów rodzinnych.

W analizie przyjmujemy m oraz k jako zmienne parametry. Nawet jeśli użytkownik posiada wielu znajomych (czyli ma wysoki stopień w grafie), ograniczenie k sprawia, że może objąć współdzieleniem tylko określoną maksymalną liczbę osób, natomiast ograniczenie m wymusza, by grupy nie były zbyt małe. Parametry te modelują zarówno ograniczenia techniczne narzucane przez dostawców usług, jak i czynniki społeczne, takie jak opłacalność czy gotowość do współdzielenia subskrypcji. W konsekwencji grupy współdzielenia odwzorowują typowe sytuacje, w których w praktyce licencje grupowe są wykorzystywane.

2.3.2 Struktura kosztów i modele cenowe

W analizowanym problemie istotnym elementem jest sposób odwzorowania polityki cenowej dostawców usług. Różne plany subskrypcyjne charakteryzują się nie tylko innym kosztem, ale również odmiennym zakresem liczby użytkowników k, którzy mogą korzystać z jednej licencji. Dodatkowo wprowadzane jest ograniczenie dolne m, które definiuje minimalną liczbę użytkowników planu. Mimo że w rzeczywistości takie ograniczenia nie są formalnie narzucane, w prezentowanym modelu pełnią ważną rolę, gdyż odzwierciedlają ekonomiczny sens wyboru licencji grupowych. W praktyce bowiem korzystanie z planu grupowego przez jedną osobę jest nieopłacalne. Przykładowo, dla planów typu Duo czy Family naturalne jest przyjęcie $m = 2$, mimo że technicznie możliwe byłoby używanie takiej subskrypcji w pojedynkę. Rodzinę typów licencji zdefiniowaną we wzorze (2.1) stosujemy tutaj do opisu struktury kosztów i ograniczeń dostępnych planów. W testach syntetycznych wartości parametrów licencji dobierane są eksperymentalnie, co pozwala badać zachowanie algorytmów w różnych wariantach cenowych i strukturalnych. W testach na danych rzeczywistych wykorzystywane są faktyczne ceny subskrypcji. Dla wariantu teoretycznego odpowiadającego dominacji rzymskiej koszty normalizowane są względem licencji indywidualnej, co umożliwia powiązanie modelu z klasycznymi zagadnieniami teorii grafów. Przykłady zestawiono w Tabeli 2.1.

W przypadku usług rzeczywistych koszty planów grupowych są znacznie niższe niż suma odpowiadających im licencji indywidualnych. Na przykład plan Spotify Family pozwala na współdzielenie subskrypcji przez maksymalnie sześć osób przy koszcie $37{,}99$ PLN, co czyni go zdecydowanie bardziej opłacalnym od planu indywidualnego ($23{,}99$ PLN). Podobna sytuacja występuje w przypadku Duolingo Super Family.

W modelu teoretycznym opartym na dominacji rzymskiej koszty są podane w jednostkach względnych. Licencja indywidualna ma koszt 1.0, a licencja grupowa koszt 2.0, co odpowiada klasycznemu przypisaniu wag w problemie dominacji rzymskiej. W ramach przeprowadzanych eksperymentów etykieta „2" będzie zmieniała swoją wartość jako wielokrotność kosztu ceny indy-

Tabela 2.1: Przykładowe modele licencji dla usług rzeczywistych i modelu teoretycznego

Typ licencji	**Koszt** c_t	**Min** m_t	**Max** k_t
Spotify (ceny w PLN)			
Individual	23,99	1	1
Duo	30,99	2	2
Family	37,99	2	6
Netflix (ceny w PLN)			
Basic	33,00	1	1
Standard	49,00	1	2
Premium	67,00	1	4
Duolingo Super (ceny w PLN)			
Individual	13,99	1	1
Family	29,17	2	6
Model teoretyczny (koszty znormalizowane)			
Solo	1,0	1	1
Group	2,0	2	999999

Źródło: opracowanie własne modelu teoretycznego oraz [6], [7], [8].

widualnej, co pozwala na analizę różnych wariantów tego zagadnienia.

2.3.3 Zakup jednoczesny i sekwencyjny

W najprostszym wariancie zakłada się, że wszystkie decyzje zakupowe zapadają w tym samym momencie. Umożliwia to globalną optymalizację i stanowi punkt odniesienia dla analiz teoretycznych. W praktyce jednak proces współdzielenia licencji jest bardziej dynamiczny. Użytkownicy dołączają do planów w różnych chwilach, część początkowo wybiera licencje indywidualne, a dopiero później postanawia zakupić subskrypcję grupową. Takie zmiany licencji mogą również wynikać z równoczesnej zmiany i ewolucji sieć społecznościowej w czasie rzeczywistym. Relacje znajomości mogą zanikać, powstawać mogą nowe połączenia, a liczba aktywnych użytkowników zmienia się w czasie.

Takie sytuacje można modelować jako proces sekwencyjny, w którym w kolejnych krokach przydzielane są nowe licencje przy uwzględnieniu bieżącej struktury grafu. Prowadzi to do odmiennych trudności niż w przypadku wariantu uproszczonego, czyli jednoczesnego. Rozwiązanie optymalne globalnie może okazać się nieosiągalne, ponieważ wcześniejsze decyzje oraz zmiany w strukturze sieci ograniczają przestrzeń dostępnych opcji w późniejszych etapach.

W pracy przeanalizowane zostały konsekwencje tego typu dynamicznych zmian takie jak stabilność i trwałość powstałych grup, wpływ zmian w grafie na opłacalność wcześniejszych decyzji, a także mechanizmy sprzyjające koordynacji i adaptacji użytkowników. Mimo że główny nacisk położony jest na wariant jednoczesny, wariant sekwencyjny stanowi istotne rozszerzenie modelu i lepiej odzwierciedla rzeczywiste warunki funkcjonowania sieci społecznościowych.

3. ZWIĄZEK Z DOMINOWANIEM W GRAFACH

3.1 Dominowanie - podstawowe definicje

Problematyka dominowania w grafach jest dobrze zbadana i szeroko opisana w literaturze [9]. W szczególności znane są klasyczne definicje zbioru dominującego oraz liczby dominowania, które stanowią punkt odniesienia dla dalszych analiz. Niech $G = (V, E)$ będzie grafem nieskierowanym. Zbiorem dominującym nazywamy podzbiór $D \subseteq V$ taki, że każdy wierzchołek spoza D ma co najmniej jednego sąsiada w D:

$$\forall v \in V \setminus D \;\; \exists\, u \in D : \{u, v\} \in E, \tag{3.1}$$

gdzie:

- V - zbiór wierzchołków grafu,
- E - zbiór krawędzi grafu,
- $D \subseteq V$ - zbiór dominujący,
- $u, v \in V$ - wierzchołki grafu,
- $\{u, v\} \in E$ - krawędź łącząca wierzchołki u i v.

Minimalną liczność zbioru dominującego oznaczamy symbolem $\gamma(G)$ i nazywamy liczbą dominowania:

$$\gamma(G) = \mathsf{min}\{\, |D| : D \subseteq V \,\}, \tag{3.2}$$

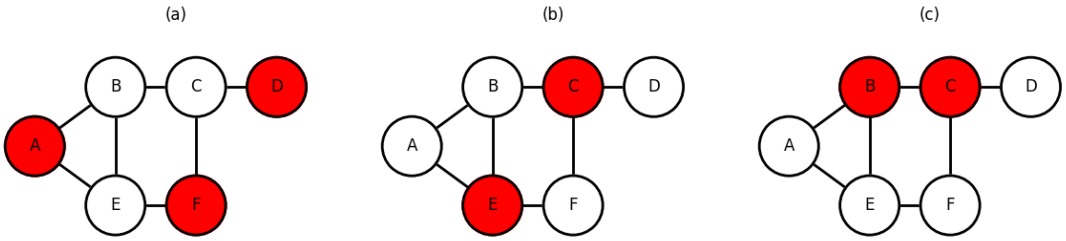
gdzie:

- $\gamma(G)$ - liczba dominowania grafu G,
- D - zbiór dominujący w grafie G,
- V - zbiór wierzchołków grafu,
- $|D|$ - liczność zbioru D.

Z punktu widzenia rozważanego problemu zakupu licencji, podstawowe założenie jest następujące: jeśli potraktujemy osoby kupujące licencje jako zbiór D, a krawędzie grafu jako relacje umożliwiające udostępnianie licencji, wówczas warunek dominowania opisany równaniem (3.1) określa sytuację, w której każdy użytkownik spoza D ma znajomego z D, a zatem uzyskuje dostęp do usługi. Gdyby wszystkie licencje były identyczne i pozwalały obsłużyć dowolną liczbę sąsiadów, minimalizacja kosztu sprowadzałaby się do wyznaczenia liczby dominowania (3.2). W praktyce jednak występują różne typy licencji oraz limity współużytkowników, co czyni problem znacznie bardziej złożonym.

Należy zauważyć, że minimalny zbiór dominujący nie musi być jednoznaczny. Wynika to bezpośrednio z definicji liczby dominowania (3.2), która określa jedynie minimalną liczność zbioru dominującego, a nie jego unikalność. Często bywa tak, że graf ma wiele różnych zbiorów domi-

nujących o rozmiarze równym $\gamma(G)$, tak jak przedstawiono na Rys. 3.1, gdzie dwa różne zbiory wierzchołków są minimalne. Problem znajdowania $\gamma(G)$ jest jednak dobrze określony i należy do klasy problemów NP-trudnych. Jego wersja decyzyjna sformułowana jest następująco: dla zadanego grafu G oraz liczby całkowitej k pytamy, czy istnieje zbiór dominujący o rozmiarze co najwyżej k. Jest to klasyczny problem NP-zupełny [10, 11, 12]. Oznacza to, że najprawdopodobniej (przy założeniu $P \neq NP$) nie istnieje algorytm wielomianowy rozwiązujący ten problem w ogólności. W praktyce stosuje się zatem algorytmy przybliżone lub ogranicza analizę do specjalnych klas grafów, dla których problem staje się prostszy. Znane są między innymi efektywne algorytmy zachłanne, które zapewniają rozwiązanie przybliżone z gwarantowanym współczynnikiem. Przykładem jest prosty algorytm zachłanny wybierający kolejno dominujące wierzchołki, osiągający przybliżenie rzędu $O(\ln n)$.



Rysunek 3.1: Przykładowe zbiory dominujące (wyróżnione na czerwono). **(a)** Zbiór dominujący o mocy 3 . **(b)**-**(c)** Dwa różne minimalne zbiory dominujące o mocy 2, zatem $\gamma(G) = 2$. Każdy wierzchołek nieczerwony ma sąsiada czerwonego.

Wiadomo jednak, że nie da się w ogólności przekroczyć bariery logarytmicznej. Problem zbioru dominującego jest APX-trudny, a dokładniej log-APX-zupełny [11]. Co więcej, nawet na bardzo ograniczonych grafach, np. grafach o maksymalnym stopniu 3 (grafy kubiczne), problem pozostaje NP-trudny i APX-zupełny [13]. Berman i Fujito (1999) wykazali m.in. NP-trudność pewnych wariantów dominacji w grafach o ograniczonym stopniu [14], co potwierdza, że zasadnicza trudność problemu dominowania jest obecna już w stosunkowo prostych strukturach.

3.2 Dominowanie rzymskie a licencje grupowe

Dominowanie rzymskie to wariant problemu dominacji, w którym każdemu wierzchołkowi przypisuje się jedną z trzech wartości: 0, 1 lub 2. Wierzchołek o wartości 1 pokrywa wyłącznie samego siebie, natomiast wierzchołek o wartości 2 pokrywa zarówno siebie, jak i wszystkich swoich sąsiadów. Wymaga się przy tym, aby każdy wierzchołek o wartości 0 był pokryty przez co najmniej jednego sąsiada oznaczonego wartością 2. Formalnie, funkcja dominowania rzymskiego na grafie $G = (V, E)$ to funkcja $f : V \to \{0, 1, 2\}$, spełniająca warunek, że dla każdego wierzchołka v z $f(v) = 0$ istnieje sąsiad $u \in V$ taki, że $f(u) = 2$ [15]. Minimalizacja sumy wartości $f(v)$ po wszystkich $v \in V$ prowadzi do zdefiniowania tzw. liczby dominowania rzymskiego grafu, oznaczanej $\gamma_R(G)$.

Terminologia i metafora związana z dominacją rzymską wywodzą się z legendy o obro-

nie granic imperium rzymskiego. Zakładano w niej, że w każdej osadzie można umieścić pewną liczbę jednostek wojskowych. Osada z dwiema jednostkami była w stanie bronić się samodzielnie i jednocześnie wysłać wsparcie do sąsiedniej osady. Lokacja z jedną jednostką potrafiła bronić tylko siebie, a miejscowości pozbawione jednostek militarnych wymagała ochrony z zewnątrz. W tej metaforze graf reprezentuje system osad i połączeń między nimi, a etykiety 0, 1 i 2 odpowiadają decyzjom o rozmieszczeniu wojsk. Koncepcję dominowania rzymskiego wprowadzili do teorii grafów Cockayne i współpracownicy w 2004 roku [16].

W kontekście problemu optymalnego zakupu licencji w grafach reprezentujących sieci społecznościowe interpretacja jest bezpośrednia. Wartość $f(v) = 2$ odpowiada użytkownikowi v, który nabywa licencję grupową i zapewnia dostęp zarówno sobie, jak i co najmniej jednemu ze swoich sąsiadów. Wartość $f(v) = 1$ oznacza użytkownika posiadającego licencję indywidualną, pokrywającą wyłącznie jego samego. Natomiast $f(v) = 0$ reprezentuje użytkownika bez własnej licencji, który musi polegać na pokryciu przez sąsiada z wartością 2. Warunek dominowania rzymskiego, zgodnie z którym każdy wierzchołek z etykietą 0 ma sąsiada z etykietą 2, gwarantuje dokładnie to, co w naszym modelu jest wymagane: każdy użytkownik bez licencji ma znajomego z licencją grupową, który może podzielić się z nim dostępem. W ten sposób każda funkcja $f : V \to \{0, 1, 2\}$ spełniająca warunki dominowania rzymskiego wyznacza dopuszczalną strategię licencyjną w rozważanej sieci.

Waga funkcji dominowania rzymskiego definiowana jest jako $w(f) = \sum_{v \in V} f(v)$, czyli suma przypisanych wartości. Liczba dominowania rzymskiego $\gamma_R(G)$ to najmniejsza możliwa waga funkcji dominowania rzymskiego dla grafu G. Jeśli przyjmiemy, że koszt licencji indywidualnej wynosi 1, a grupowej 2, to minimalizacja kosztu w naszym problemie pokrywa się z zagadnieniem znalezienia funkcji dominowania rzymskiego o minimalnej wadze. Cel minimalizacji sumarycznego kosztu $C = |I| + 2|G|$ jest zatem równoważny minimalizacji $w(f)$, gdy utożsamimy $|I|$ z liczbą wierzchołków o etykiecie 1, a $|G|$ o etykiecie 2. Należy jednak podkreślić, że pełna równoważność z dominowaniem rzymskim zachodzi tylko wtedy, gdy liczba sąsiadów dowolnego wierzchołka nie przekracza maksymalnej liczby użytkowników dopuszczonych w planie grupowym. Jak wspomniano wcześniej przy omawianiu parametrów m i k, nasze ujęcie wprowadza dodatkowe ograniczenie pojemności - wierzchołek o wartości 2 może pokrywać jedynie ograniczoną liczbę sąsiadów. Problem optymalnego zakupu licencji jest więc uogólnieniem dominowania rzymskiego, dostosowanym do praktycznych limitów występujących w planach subskrypcyjnych.

W przypadku innych modeli cenowych, dominacja rzymska stanowi nadal użyteczną metaforę, choć nie oddaje w sposób wystarczający struktury kosztów. Gdy $p \neq 2$, możemy rozważyć ogólniejsze przypisania wag, w których etykieta odpowiada rzeczywistemu kosztowi planu. Przykładowo, gdy $p = 3$, to licencja grupowa ma koszt równy trzem jednostkom, co nie mieści się w klasycznym schemacie $\{0, 1, 2\}$. W literaturze zaproponowano rozszerzenia pozwalające modelować takie sytuacje, m.in. k-dominację rzymską, gdzie dopuszcza się wartości $\{0, 1, \ldots, k\}$ [17], czy też dominację rzymską z wagami [18]. Warianty te pozwalają odwzorować przypadki, w których dostępne są plany o różnych kosztach i różnej pojemności, np. licencja droższa, ale

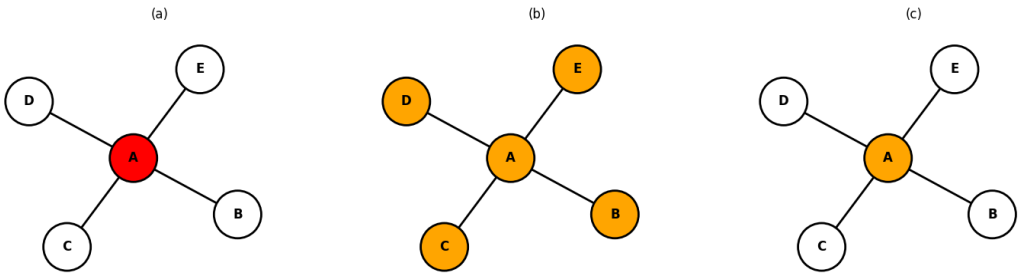
umożliwiająca współdzielenie w większej grupie. W tym sensie uogólnienia dominowania rzymskiego są bliższe rzeczywistemu problemowi optymalizacji kosztów licencji niż klasyczna wersja ograniczona do wartości 0, 1 i 2.

Na potrzeby tej pracy nie jest jednak konieczne wchodzenie w szczegółowe odmiany dominowania rzymskiego. Wystarczy zauważyć, że w analizowanym modelu schemat pozostaje taki sam jak w klasycznej wersji, a jedyną różnicą jest koszt przypisywany wierzchołkom o etykiecie 2. W zależności od przyjętego modelu cenowego wartość ta może wynosić $p = 2$, ale równie dobrze $p = 1{,}5$ czy $p = 3$. Konstrukcja funkcji dominowania rzymskiego oraz sam podział na etykiety $\{0, 1, 2\}$ nie ulega zmianie. Innymi słowy, dominacja rzymska dostarcza prostego i intuicyjnego opisu sytuacji.

Prosty przykład zastosowania dominowania rzymskiego można przedstawić na grafie typu gwiazda. Centralny wierzchołek A jest połączony z liśćmi B, C, D, E. Najlepszą strategią jest, aby A wykupił licencję grupową i udostępnił ją wszystkim swoim sąsiadom. W modelu oznacza to $f(A) = 2$, a dla każdego liścia $f(B) = f(C) = f(D) = f(E) = 0$. Warunek dominowania rzymskiego jest spełniony, ponieważ każdy liść o wartości 0 ma sąsiada A o wartości 2. Waga funkcji wynosi wówczas $w(f) = 2$ - odpowiada to kosztowi jednej licencji grupowej obsługującej całą pięcioosobową grupę.

Dla porównania można rozważyć inne strategie. Jeśli każdy wierzchołek kupiłby licencję indywidualną ($f = 1$), waga wyniosłaby 5 - koszt pięciu osobnych licencji. Jeśli centralny wierzchołek kupiłby tylko licencję indywidualną ($f(A) = 1$), a liście pozostałyby bez dostępu, warunek nie byłby spełniony - żaden liść nie miałby sąsiada o wartości 2.

Rysunek 3.2 ilustruje ten przykład i pokazuje, jak dominowanie rzymskie wskazuje optymalny wybór użytkownika pełniącego rolę lidera w centrum gwiazdy. W większych grafach kandydatów do roli właścicieli licencji grupowych jest więcej - ich odpowiedni dobór prowadzi już do właściwego problemu optymalizacji dominowania rzymskiego.



Rysunek 3.2: Kolory: biały = 0, pomarańczowy = 1, czerwony = 2. **(a)** Przypisanie optymalne (koszt 2). **(b)** Wszyscy z $f = 1$ (koszt 5). **(c)** $A = 1$, liście $f = 0$ (niespełniony warunek).

3.3 Złożoność obliczeniowa problemu

Dokładne dowody znajdują się w literaturze - np. pokazano NP-zupełność problemu dominowania rzymskiego poprzez redukcję z problemu pokrycia wierzchołków lub zbioru dominującego [19]. W kontekście naszego modelu oznacza to, że optymalne wyznaczenie użytkowników kupu-

jących poszczególne licencje jest obliczeniowo trudne dla dużych sieci. Innymi słowy, nie istnieje znany algorytm wielomianowy, który gwarantowałby znalezienie rozwiązania minimalnego kosztu dla dowolnej struktury grafu znajomości - w najgorszym przypadku liczba kombinacji rośnie wykładniczo wraz z liczbą wierzchołków.

Konsekwencją NP-trudności jest także brak prostego schematu aproksymacyjnego dla problemu minimalizacji kosztów licencji. Ponieważ problem dominowania (minimalnego zbioru dominującego) jest APX-zupełny (dla konkretnych rodzajów grafów) [11], nie ma wielomianowego schematu aproksymacji (PTAS) gwarantującego dobre przybliżenie. Dla naszego problemu, który jest uogólnieniem dominowania, można oczekiwać podobnych ograniczeń - prawdopodobnie nie da się znaleźć w czasie wielomianowym rozwiązań bliższych optimum niż o czynnik $O(\ln n)$ w najgorszym przypadku. Proste heurystyki zachłanne mogą jednak dawać przyzwoite wyniki. Na przykład, heurystyka wybierająca iteracyjnie wierzchołek, który pokrywa najwięcej jeszcze niepokrytych sąsiadów (i dająca mu licencję grupową), jest jedną z implementacji algorytmu zachłannego dla zbioru dominującego i osiąga współczynnik $\approx (2 + \ln \Delta)$, gdzie Δ to maksymalny stopień grafu [20]. W najgorszym razie jest to $O(\ln n)$, ale dla wielu grafów rzeczywiste wyniki są lepsze niż ta pesymistyczna granica. Ważnym faktem jest też to, że problem pozostaje trudny nawet dla grafów o niewielkich stopniach - np. wykazano, że dla grafów o stopniu mniejszym bądź równym cztery minimum dominujące jest APX-zupełne [13, 11]. To implikuje, że ograniczenie maksymalnej liczby znajomych nie czyni problemu trywialnym. W kontekście subskrypcji oznacza to, że nawet w sieci gdzie każdy zna tylko kilka osób, optymalny dobór kto z kim powinien się połączyć we wspólnej licencji nadal może wymagać złożonych obliczeń.

Biorąc powyższe pod uwagę, w dalszej części badania główny nacisk zostanie położony na podejścia algorytmiczne, które biorą pod uwagę tę złożoność. Ponieważ nie istnieje wydajny algorytm dokładny dla ogólnego przypadku, rozważymy dwutorowo: (a) zastosowanie metod dokładnych dla umiarkowanych rozmiarów oraz (b) zaprojektowanie i analiza algorytmów heurystycznych zdolnych dawać dobre rozwiązania dla większych sieci. W literaturze pojawiły się już pierwsze próby wykorzystania technik optymalizacyjnych do problemu dominacji. Przykładowo, Parra Inza i in. (2022) zaproponowali sformułowanie problemu dominującego jako ILP oraz heurystyki naprawcze dla jego rozwiązywania [21]. Takie podejście można zaadaptować do omawianego problemu, wprowadzając zmienne decyzyjne wskazujące wybór typu licencji dla każdego użytkownika i ograniczenia. Z drugiej strony, heurystyki takie jak algorytm zachłanny, algorytmy lokalnej optymalizacji czy metaheurystyki - np. algorytmy genetyczne, symulowane wyżarzanie - mogą być użyte, by szybko przeszukać przestrzeń możliwych konfiguracji licencji.

4. DANE TESTOWE

W celu przeprowadzenia szczegółowej analizy efektywności algorytmów optymalizujących zakup licencji w sieciach społecznościowych niezbędne było wykorzystanie różnorodnych danych testowych. Posłużyły do tego syntetycznie generowane grafy losowe oraz rzeczywiste fragmenty sieci społecznościowej. Pierwsza grupa stanowi kontrolowany zbiór danych sztucznych, pozwalający na symulowanie różnych scenariuszy topologicznych i analizę wpływu struktury sieci na działanie algorytmów. Druga grupa to rzeczywiste ego-sieci z platformy Facebook, umożliwiające weryfikację metod na prawdziwych danych społecznościowych.

Niniejszy rozdział koncentruje się na charakterystyce wykorzystanych danych, natomiast ich praktyczne zastosowanie zostanie szczegółowo przedstawione w części poświęconej opisowi przeprowadzonych eksperymentów.

4.1 Grafy syntetyczne

Do generowania danych syntetycznych wykorzystano trzy standardowe modele grafów: Erdős–Rényi (ER), Barabási–Albert (BA) oraz Watts–Strogatz (WS). W eksperymentach wykorzystywano kilka odrębnych konfiguracji rozmiaru w zależności od scenariusza eksperymentalnego:

- **Benchmark statyczny** (rozdz. 6) obejmował grafy o $n \in \{20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 300, 600, 1000\}$, generowane po trzy próbki na rozmiar.
- **Symulacje dynamiczne** (rozdz. 7) korzystały z mniejszego wachlarza wielkości: dla mutacji syntetycznych $n \in \{20, 40, 80, 160, 320, 640\}$, natomiast warianty realistyczne (pref_triadic, pref_pref, rand_rewire) operowały odpowiednio na zestawach $\{40, 80, 160, 320\}$ oraz $\{40, 80, 160, 320, 640\}$.
- **Rozszerzenia taryfowe** (rozdz.8) w wariancie statycznym analizowano na rozmiarach $n \in \{20, 50, 100, 200\}$, natomiast część dynamiczna korzystała z $n \in \{40, 80, 160\}$ przy krótszym horyzoncie czasowym.

4.1.1 Model Erdős–Rényi - klasyczne grafy przypadkowe

Pierwszym rozważanym modelem jest klasyczny losowy graf Erdős–Rényi (ER) zaproponowany przez Erdős'a i Rényi'ego w 1959 roku [22]. W modelu tym rozpatruje się zbiór n wierzchołków, a każda z $\binom{n}{2}$ potencjalnych krawędzi pojawia się niezależnie z prawdopodobieństwem p. Parametrami modelu są więc n oraz p. W używanej implementacji zaadaptowano właśnie ten wariant $G(n, p)$.

Model ER stanowi istotny punkt odniesienia jako najprostszy model sieci pozbawiony struktury społecznościowej. Motywacją uwzględnienia go w testach jest możliwość porównania działania algorytmów na zupełnie przypadkowych sieciach z ich działaniem na bardziej uporządkowanych grafach takich jak grafy skalowane, małego świata oraz rzeczywiste. Choć prawdziwe sieci społecznościowe odbiegają od założeń pełnej losowości, mają zwykle wyższy poziom klaste-

ryzacji węzłów i nierównomierny rozkład stopni, to jednak model $G(n,p)$ stanowi istotną podstawę porównawczą.

Z punktu widzenia właściwości, grafy ER cechują się stosunkowo niskim średnim współczynnikiem klasteryzacji. Oczekiwana wartość współczynnika klasteryzacji jest równa p dla wystarczająco dużego n oraz dla dostatecznie dużego p istnieje możliwość powstania jednej gigantycznej składowej spójnej. Istnieje znana granica perkolacji, gdy p przekroczy około $\frac{\ln n}{n}$, graf $G(n,p)$ jest z dużym prawdopodobieństwem spójny - poniżej tego progu sieć rozpada się na wiele komponentów [22]. Gęstość grafu, rozumiana jako odsetek istniejących krawędzi w stosunku do wszystkich możliwych, wynosi w tym modelu w przybliżeniu p. Przykładowo dla $p = 0.1$ graf będzie miał ok. 10% maksymalnej liczby krawędzi. Rozkład stopni w modelu ER ma charakter dwumianowy, a w granicy dużego n zbiega do rozkładu Poissona. Oznacza to, że w grafach tych nie występują węzły o niezwykle wysokich stopniach (tzw. huby), które są charakterystyczne dla wielu rzeczywistych sieci społecznościowych. W konsekwencji model ER nie oddaje wielu kluczowych właściwości takich sieci - stanowi jednak użyteczny model kontrolny, pozbawiony zjawisk typu mały świat czy skalowość, dzięki czemu można wyraźnie uwypuklić wpływ tych cech w innych modelach.

4.1.2 Model Barabási–Albert - sieci bezskalowe

Drugim wykorzystanym generatorem jest model Barabási–Albert (BA), wprowadzony przez Barabási'ego i Alberta w 1999 roku [23]. Model BA pozwala generować grafy o strukturze bezskalowej, których rozkład stopni wierzchołków przyjmuje postać potęgową. Tego typu sieci charakteryzują się istnieniem niewielkiej liczby wierzchołków o bardzo wysokim stopniu (tzw. hubów) oraz wielu wierzchołków o małym stopniu - jest to cecha obserwowana w wielu rzeczywistych sieciach, w tym społecznościowych. W przypadku sieci społecznościowych odnosi się to do tego, że niektórzy użytkownicy mogą mieć tysiące znajomych/obserwujących, podczas gdy większość ma ich kilkudziesięciu lub mniej.

Parametrem wejściowym modelu Barabási–Albert jest przede wszystkim n - docelowa liczba węzłów w grafie - oraz m - liczba krawędzi, jakie dodaje każdy nowy węzeł. Procedura generowania rozpoczyna się od małego grafu startowego (np. klika złożona z m wierzchołków, aby zapewnić początkową spójność). Następnie dodaje się kolejno nowe wierzchołki; każdy nowy węzeł łączy się z m już istniejącymi wierzchołkami, przy czym prawdopodobieństwo połączenia z danym istniejącym węzłem jest proporcjonalne do jego bieżącego stopnia (tzw. reguła preferencyjnego łączenia, ang. preferential attachment). W efekcie bogaci stają się bogatsi, czyli wierzchołki, które zyskały wiele połączeń na wcześniejszych etapach, mają większą szansę zdobyć kolejne połączenia, co prowadzi do wykładniczego rozkładu stopni.

Motywacją użycia modelu BA było odzwierciedlenie w danych testowych właściwości często spotykanej w sieciach społecznościowych i sieciach informacji - silnego zróżnicowania w stopniach węzłów. Dzięki grafom BA można przetestować algorytmy pod kątem radzenia sobie z obecnością hubów oraz z rozkładem stopni o ciężkim ogonie (ang. heavy-tailed distribution). Pojęcie ciężkiego ogona oznacza, że prawdopodobieństwo wystąpienia wierzchołków o bardzo dużym

stopniu maleje stosunkowo wolno - w efekcie w sieci, obok wielu węzłów o niskim stopniu, pojawia się również pewna liczba hubów o ekstremalnie wysokim stopniu. Zjawisko to odróżnia sieci bezskalowe od np. grafów ER, w których prawdopodobieństwo pojawienia się węzłów o bardzo dużej liczbie sąsiadów jest znikome.

Grafy generowane modelem BA mają z reguły jedną spójną komponentę. Przy założeniu, że graf startowy jest spójny i $m \geq 1$, każdy nowy wierzchołek dołącza do istniejącej struktury, więc sieć pozostaje spójna, a średni stopień w takim grafie wynosi około $2m$. Stąd gęstość grafu BA maleje wraz ze wzrostem n. Dla dużych grafów jest ona rzędu $\frac{2m}{n}$, co oznacza, że grafy te są rzadkie. W przeciwieństwie do modelu ER, współczynnik skupienia grafów BA nie jest determinowany przez pojedynczy parametr w oczywisty sposób - klasyczny model BA generuje sieci o stosunkowo niskim średnim clusteringu (niższym niż obserwowany w rzeczywistych sieciach społecznościowych), ponieważ nowe połączenia tworzone są głównie z hubami, co sprzyja tworzeniu gwiazd zamiast trójkątów. Istnieją modyfikacje modelu BA dodające mechanizmy triadycznego dosłączania, które zwiększają clustering - jednak w czystej postaci model BA zazwyczaj skutkuje średnim współczynnikiem skupienia malejącym wraz z rozmiarem grafu. Niemniej jednak, nawet przy relatywnie niskim clusteringu, grafy BA zachowują własność małych średnich odległości. Powyższe cechy sprawiają, że grafy BA stanowią przydatny model testowy - oddają one istnienie hubów i krótkich odległości jak w wielu sieciach społecznościowych, choć nie odwzorowują silnego grupowania lokalnego.

4.1.3 Model Watts–Strogatz - graf małego świata

Trzecim fundamentalnym modelem wykorzystanym w pracy jest model Watts–Strogatz (WS), opisany przez Wattsa i Strogatza w 1998 roku [24]. Umożliwia on generowanie grafów małego świata (small-world networks), które łączą w sobie dwie istotne cechy: wysoki współczynnik skupienia (podobny do obserwowanego w sieciach regularnych) oraz niską średnią odległość (podobnie jak w grafach losowych). Model ten odzwierciedla fakt, że w sieciach społecznościowych często występują silnie zżyte grupy znajomych, a jednocześnie dowolne dwie osoby są połączone relatywnie krótką ścieżką znajomości.

Generacja grafu WS wymaga dostarczenia mu trzech parametrów o liczbie wierzchołków, stopniu każdego wierzchołka w początkowej regularnej strukturze oraz prawdopodobieństwa przepięcia krawędzi. Procedura rozpoczyna się od utworzenia grafu regularnego, gdzie każdy wierzchołek jest połączony z k najbliższymi sąsiadami w pierścieniu (tj. tworzymy pierścień z n węzłów, a następnie każdy węzeł łączymy z $\frac{k}{2}$ następnymi i $\frac{k}{2}$ poprzednimi na pierścieniu, zakładając dla uproszczenia, że k jest parzyste). Tak powstała sieć ma wysokie lokalne skupienie - węzły sąsiadujące na pierścieniu tworzą krótkie kliki. Jednocześnie początkowa średnia odległość jest stosunkowo duża bo graf tuż przed modyfikacją krawędzi ma strukturę pierścienia, więc dystans między węzłami oddalonymi na pierścieniu jest znaczny.

Następnie, w modelu WS wprowadza się losowe przepięcia. Dla każdej krawędzi łączącej węzeł z jednym z $\frac{k}{2}$ najbliższych sąsiadów w pierścieniu. Dokonuje się, z prawdopodobieństwem

p, przepięcia jednego końca tej krawędzi do losowo wybranego innego wierzchołka. Przepięcie polega na usunięciu oryginalnej krawędzi i dodaniu nowej krawędzi łączącej dany węzeł z innym losowym węzłem. W wyniku tych losowych przepięć przy zachowaniu większości lokalnych połączeń pierścienia otrzymujemy graf, który dla małych p wciąż ma wysoki współczynnika klasteryzacji, ale jednocześnie kilka losowych skoków znacząco skraca średnie odległości w sieci. Dla umiarkowanych wartości p (np. $p \approx 0.01$ czy 0.1) sieć uzyskuje bardzo małą średnią odległość - zbliżoną do grafów losowych - podczas gdy clustering pozostaje o rząd wielkości wyższy niż w grafie Erdős–Rényi o porównywalnej gęstości.

W kontekście modelowania sieci społecznościowych, generator dla modelu WS dodano w celu odzwierciedlenia właściwości, których brakuje modelowi BA - mianowicie wysokiego lokalnego współczynnika klasteryzacji. Sieci społecznościowe cechują się tym, że znajomi często znają się nawzajem, tworząc kliki znajomych. Model WS pozwala symulować taką sytuację i sprawdzić, jak algorytmy radzą sobie np. z wykrywaniem społeczności czy zjawisk rozprzestrzeniania się informacji w warunkach silnego grupowania. Parametr k decyduje o początkowej gęstości połączeń lokalnych - większe k to więcej krawędzi lokalnych (każdy węzeł ma początkowo k sąsiadów), a zatem wyjściowo wyższy współczynnik klasteryzacji i gęstość. Parametr p natomiast kontroluje losowość grafu. Przykładowo dla $p = 0$ otrzymujemy graf regularny, zaś dla $p = 1$ graf staje się w dużej mierze losowy. W praktycznych zastosowaniach interesujący jest zakres p między 0 a 1, gdzie pojawia się mały świat.

Grafy WS generowane do testów miały parametry dobrane w taki sposób, aby możliwie dobrze odwzorowywać cechy typowe dla niedużych sieci społecznościowych. Uzyskiwane w ten sposób sieci charakteryzowały się relatywnie niską gęstością, ale jednocześnie wysokim współczynnikiem klasteryzacji, znacznie przewyższającym wartości obserwowane w losowych grafach ER o podobnej gęstości. Dzięki temu w grafach WS obecne są realistyczne zgrupowania lokalne, odpowiadające typowym kręgom znajomych w sieciach społecznościowych. Co istotne, sieci te z reguły pozostają spójne - niemal wszystkie wierzchołki należą do jednej dużej komponenty, a ewentualne izolowane węzły pojawiają się jedynie sporadycznie przy skrajnych ustawieniach parametrów. Taka struktura sprawia, że model WS stanowi dobre środowisko testowe.

4.2 Grafy rzeczywiste

Drugim zestawem danych testowych są rzeczywiste grafy pochodzące z sieci społecznościowej Facebook, a dokładniej zbiór Facebook Ego Network udostępniony w ramach Stanford Network Analysis Project (SNAP) [25]. Dane te zostały zebrane w 2012 roku przez J. McAuley i J. Leskovca z Uniwersytetu Stanforda w ramach badań nad automatycznym wykrywaniem kręgów społecznościowych [26]. Zbiór zawiera dziesięć tzw. ego-sieci - sieci ego-centrycznych poszczególnych użytkowników Facebooka, pozyskane za zgodą uczestników przy użyciu dedykowanej aplikacji badawczej działającej w ramach platformy Facebook. Ego-sieć to sieć społeczna z perspektywy pojedynczego użytkownika zwanego ego - węzłami są ego oraz wszystkie jego bezpośrednie znajome osoby, zaś krawędzie reprezentują relacje znajomości pomiędzy tymi znajomymi.

W udostępnionych danych każda z dziesięciu sieci odpowiada innemu użytkownikowi i zawiera wyłącznie jego znajomych oraz powiązania między nimi. Węzeł ego nie jest jawnie ujęty jako wierzchołek w grafie i można go traktować jako ukrytą centralną jednostkę łączącą wszystkich znajomych. Innymi słowy, graf zapisany w pliku `X.edges` dotyczy tylko znajomych użytkownika X i relacji między nimi - sam X nie pojawia się w pliku jako węzeł.

4.2.1 Struktura danych

Każda ego-sieć udostępnione przez SNAP zapisana jest w osobnych plikach tekstowych, których nazwa odpowiada identyfikatorowi *ego* (np. `0.edges`, `0.circles`, `0.feat`, `0.egofeat` dla *ego* o ID=0). Struktura danych jest następująca:

Plik .edges - lista krawędzi w grafie znajomych danego ego. Każdy wiersz zawiera dwie liczby - identyfikatory dwóch różnych znajomych ego, między którymi istnieje relacja koleżeńska. Krawędzie te są nieskierowane. Ważną cechą jest to, że plik `.edges` nie zawiera połączeń od ego do jego znajomych - wierzchołek ego w ogóle nie występuje w tym pliku. Oznacza to, że rzeczywista sieć ego (gdyby uwzględnić w niej węzeł ego) miałaby dodatkowo krawędź łączącą ego z każdym z pojawiających się znajomych, jednak tych połączeń tutaj nie zapisano (są one domyślne - zakładamy, że ego jest połączone ze wszystkimi swoimi znajomymi). Pominięcie węzła ego jest zabiegiem celowym, pozwalającym skupić się na relacjach wewnątrz kręgów znajomych. Konsekwencją tego jest często podział grafu znajomych na kilka komponentów - jeśli ego ma różne grupy znajomych wzajemnie się nieznających, to w pliku `.edges` każda taka grupa stanowi osobną składową spójną, jako jedyny łącznik, został usunięty z grafu). Przykładowo, w ego-sieci `0.edges` znajomi tworzą 5 odrębnych komponentów spójnych. Oznacza to, że użytkownik o ID 0 miał około pięć niezależnych grup znajomych niepowiązanych ze sobą - dopiero poprzez jego osobę stawały się one pośrednio połączone.

Plik .circles - zestaw kręgów znajomych zdefiniowanych przez użytkownika. Każdy wiersz pliku reprezentuje jeden krąg towarzyski. Wiersz rozpoczyna się od nazwy kręgu - jednak w udostępnionych danych nazwy te zostały zanonimizowane lub pominięte, więc w praktyce każdy wiersz zaczyna się od identyfikatora kręgu albo pustej nazwy, po czym następuje lista ID użytkowników należących do tego kręgu. Kręgi mogą częściowo się pokrywać i nie muszą stanowić rozłącznych społeczności w sensie grafu - są to raczej dodatkowe metadane od ego, opisujące jak kategoryzuje on swoich znajomych. Informacje te mogą być cenne pomocniczo, np. w oryginalnej pracy McAuley'ego i Leskovca posłużyły do oceny algorytmów automatycznie wykrywających społeczności [26].

Plik .feat - macierz cech atrybutów przypisanych do znajomych ego. Każdy wiersz odpowiada jednemu znajomemu i zawiera wektor wartości cech tej osoby. Cechy te mogą obejmować informacje z profilu Facebooka (np. miejsce pracy, szkoła, zainteresowania itp.). W udostępnionym zbiorze wartości atrybutów zostały zanonimizowane - nie znamy dokładnego znaczenia poszczególnych cech, jedynie ich binarne wartości (1 - użytkownik posiada daną cechę, 0 - nie posiada). Istnieje także plik `.featnames` zawierający oryginalne nazwy cech, ale w przypadku Facebooka

nazwy te również zostały zanonimizowane (np. zamiast ”szkoła: Uniwersytet Stanford” pojawia się anonimowa cecha 57). W niniejszej pracy dane atrybutów nie były wykorzystywane przez algorytmy i skupiono się tylko na strukturze grafów.

Plik .egofeat - wektor cech centralnego użytkownika, w tym samym formacie co pojedynczy wiersz pliku `.feat`, odnoszący się jednak do ego. Pozwala to porównać cechy ego z cechami jego znajomych. W kontekście naszych badań plik ten również nie był bezpośrednio wykorzystywany, poza podstawową walidacją danych.

W eksperymentach wykorzystano wszystkie dziesięć ego-sieci dostępnych w zbiorze SNAP. Liczba węzłów (po pominięciu centralnego ego) wynosiła odpowiednio 53, 62, 151, 169, 225, 334, 535, 748, 787 oraz 1035. Tak szeroki zestaw umożliwił ocenę algorytmów zarówno na niewielkich, jak i ponadtysięcznych grafach, w których różnice w gęstości i strukturze społeczności są wyraźnie widoczne.

4.2.2 Szczegółowy opis danych ze zbioru SNAP

W badanym zbiorze występują zarówno niewielkie sieci liczące poniżej setki węzłów (np. ID 3980: 53 wierzchołki, 252 krawędzie), jak i bardzo duże instancje przekraczające tysiąc węzłów (ID 0: 1035 wierzchołków, ponad 26 000 krawędzi). Ogólnie rzecz biorąc, większa liczba znajomych oznacza większe zróżnicowanie strukturalne: część ego-sieci składa się z kilku gęstych społeczności, podczas gdy inne są rozproszone i tworzą liczne komponenty. Sumarycznie wszystkie dziesięć ego-sieci obejmuje 4039 unikalnych wierzchołków oraz 88 234 krawędzie [26], co dobrze oddaje skalę i złożoność sieci osobistych kontaktów.

Jak wspomniano, z powodu braku węzła ego w grafie znajomych większość ego-sieci dzieli się na więcej niż jedną składową spójną. W praktyce zazwyczaj istnieje jedna dominująca komponenta, zawierająca największą grupę wzajemnie powiązanych znajomych, oraz kilka mniejszych komponentów (np. dwu-lub kilkuosobowych grup) odpowiadających odizolowanym kręgom towarzyskim. Dla przykładu, sieć `107.edges` okazała się całkowicie spójna - wszyscy znajomi użytkownika 107 tworzyli jeden klaster połączony również ze sobą. Natomiast sieć `0.edges` miała 5 komponentów - co już sygnalizowano wcześniej. Wśród naszych analizowanych sieci: sieć `686.edges` jest spójna, sieć `414.edges` dzieli się na 2 komponenty, sieć `698.edges` na 3, a sieć `3980.edges` na 4 komponenty. Zwykle największy komponent obejmuje zdecydowaną większość wierzchołków. Taka struktura wskazuje na obecność jednego głównego kręgu znajomych, uzupełnionego kilkoma mniejszymi grupami znajomości niepowiązanych z resztą.

Ego-sieci Facebooka cechują się na ogół wysokim clusteringiem, co zgodne jest z intuicją - znajomi konkretnej osoby często znają się nawzajem. W literaturze podaje się, że globalny współczynnik klasteryzacji dla całego grafu Facebooka jest stosunkowo niski [27]. W obrębie pojedynczej ego-sieci, gdzie wszyscy rozważani ludzie są znajomymi jednego ego, współczynnik skupienia jest znacznie wyższy. Dla połączonej sieci 10 ego (4039 węzłów) średni clustering wynosił aż 0.6055, co oznacza, że dwaj losowo wybrani znajomi danego ego mieli ponad 60% szans, by również być znajomymi między sobą. W naszych mniejszych sieciach wartości te różnią się

w zależności od sieci, ale zwykle mieszczą się w przedziale 0.5-0.6 dla największego komponentu (mniejsze komponenty, np. dwuosobowe, mają clustering równy 0 lub nieokreślony). Wysoki średni współczynnik skupienia potwierdza istnienie silnych lokalnych powiązań - w grafie występuje wiele trójkątów (grup znajomych, z których każdy zna pozostałych). Przykładowo, jeśli ego posiada grupę bliskich przyjaciół ze szkoły, to prawdopodobne jest, że większość z nich zna się nawzajem, tworząc pełne podgrafy (kliki) o dużym clusteringu.

Gęstość, rozumiana jako stosunek liczby istniejących krawędzi do maksymalnej liczby krawędzi możliwej między daną liczbą wierzchołków, w ego-sieciach Facebooka jest stosunkowo niska w kategoriach bezwzględnych (co wynika z faktu, że sieci społecznościowe z natury są rzadkie). Typowe wartości w badanym zbiorze mieszczą się od kilku do kilkunastu procent: największe sieci (powyżej 700 węzłów) osiągają gęstości około 0.05, natomiast mniejsze instancje (53 i 62 węzły) przekraczają 0.15 dzięki bardziej lokalnym połączeniom. W porównaniu do grafów losowych o podobnej skali ego-sieci mają wyższy clustering (krawędzie nie są rozłożone przypadkowo, lecz skoncentrowane wewnątrz grup), natomiast same wartości gęstości pozostają rzędu pojedynczych procent. Ta rzadka natura sieci społecznościowych jest istotna z punktu widzenia testowanych algorytmów, gdyż wiele z nich ma złożoności silnie zależne od liczby krawędzi (np. operacje przeszukiwania grafu lub znajdowania struktur klikowych mogą być szybsze w grafach rzadszych).

5. METODY ALGORYTMICZNE

W tym rozdziale przedstawiono wszystkie algorytmy zastosowane w pracy. Każdy algorytm opisano wraz z jego główną ideą, parametrami oraz analizą złożoności obliczeniowej.

Algorytmy zastosowane w pracy można podzielić na trzy grupy:

1. **Metody dokładne** – gwarantują znalezienie optymalnego rozwiązania: algorytm naiwny i programowanie całkowitoliczbowe (ILP).
2. **Heurystyki konstrukcyjne** – szybkie metody budujące rozwiązanie krok po kroku: algorytm zachłanny, zbiór dominujący, algorytm losowy.
3. **Metaheurystyki** – zaawansowane metody przeszukujące przestrzeń rozwiązań: algorytm genetyczny, przeszukiwanie tabu, algorytm mrówkowy, symulowane wyżarzanie.

5.1 Metody dokładne

5.1.1 Algorytm naiwny

Algorytm naiwny to najprostsza metoda dokładna, która sprawdza wszystkie możliwe rozwiązania i wybiera najlepsze. Gwarantuje znalezienie optymalnego rozwiązania, ale działa tylko dla bardzo małych grafów.

Idea metody Algorytm systematycznie przegląda wszystkie możliwe podziały wierzchołków grafu na grupy licencyjne:

1. Generuje wszystkie partycje zbioru wierzchołków.
2. Dla każdej partycji sprawdza wszystkie możliwe przypisania licencji.
3. Weryfikuje poprawność rozwiązania.
4. Oblicza koszt i zapamiętuje najlepsze rozwiązanie.

Algorithm 1 Algorytm naiwny – przegląd wszystkich możliwych rozwiązań

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

if $|V| > 10$ **then**
    return błąd – graf zbyt duży
end if
$best\_cost \leftarrow \infty$, $best\_solution \leftarrow \emptyset$
for każda partycja $P = \{P_1, P_2, \ldots, P_k\}$ zbioru V **do**
    for każde przypisanie licencji do bloków partycji **do**
        if rozwiązanie spełnia wszystkie ograniczenia **then**
            $cost \leftarrow$ oblicz koszt rozwiązania
            if $cost < best\_cost$ **then**
                $best\_cost \leftarrow cost$, $best\_solution \leftarrow$ obecne rozwiązanie
            end if
        end if
    end for
end for
return $best\_solution$

Złożoność i zastosowanie Algorytm ma złożoność nadwykładniczą $O(c^n)$ ze względu na konieczność przeglądu wszystkich partycji zbioru wierzchołków. Z tego powodu praktyczny jest tylko dla grafów o maksymalnie 10 wierzchołkach. Używany był głównie jako punkt odniesienia do walidacji innych algorytmów.

5.1.2 Programowanie całkowitoliczbowe (ILP)

Programowanie całkowitoliczbowe to metoda dokładna, która formułuje problem jako zadanie optymalizacji liniowej ze zmiennymi całkowitoliczbowymi. Gwarantuje znalezienie optymalnego rozwiązania.

Idea metody Algorytm minimalizuje łączny koszt wszystkich utworzonych grup licencyjnych używając dwóch typów zmiennych binarnych:

- $a_{i,\ell}$ – aktywacja grupy w wierzchołku i dla typu licencji ℓ
- $x_{i,j,\ell}$ – przypisanie wierzchołka j do grupy właściciela i z licencją ℓ

Model matematyczny (dla $N[i] = N(i) \cup \{i\}$):

$$\min \quad \sum_{i \in V} \sum_{\ell \in \mathcal{L}} c_\ell \, a_{i,\ell} \qquad \text{(funkcja celu)} \tag{5.1}$$

$$\sum_{\substack{i \in V: \\ j \in N[i]}} \sum_{\ell \in \mathcal{L}} x_{i,j,\ell} = 1 \qquad \forall j \in V \quad \text{(pokrycie)} \tag{5.2}$$

$$l_\ell \, a_{i,\ell} \leq \sum_{j \in N[i]} x_{i,j,\ell} \leq u_\ell \, a_{i,\ell} \qquad \forall i \in V, \; \ell \in \mathcal{L} \quad \text{(pojemności)} \tag{5.3}$$

$$x_{i,i,\ell} = a_{i,\ell} \qquad \forall i \in V, \; \ell \in \mathcal{L} \quad \text{(właściciel w grupie)} \tag{5.4}$$

Złożoność i zastosowanie Liczba zmiennych rośnie jako $O(|V| \cdot |\mathcal{L}| \cdot \Delta)$, gdzie Δ to maksymalny stopień wierzchołka. Dla dużych i gęstych grafów czas obliczeń może być znaczny. Metoda używana jako:

- punkt odniesienia dla oceny jakości innych algorytmów,
- rozwiązanie dla małych instancji (do około 100-200 wierzchołków),
- generator górnych ograniczeń dla heurystyk.

5.2 Heurystyki konstrukcyjne

5.2.1 Algorytm zachłanny

Algorytm zachłanny to szybka heurystyka, która buduje rozwiązanie krok po kroku, w każdym kroku wybierając lokalnie najlepszą opcję. Algorytm nie gwarantuje znalezienia optymalnego rozwiązania, ale jest bardzo szybki i daje zazwyczaj dobre wyniki.

Idea metody Algorytm działa według następującej strategii:

1. Sortuje wierzchołki malejąco według liczby sąsiadów (stopnia wierzchołka)
2. Dla każdego wierzchołka sprawdza, czy może być właścicielem grupy
3. Wybiera typ licencji i rozmiar grupy tak, aby zminimalizować stosunek kosztu do rozmiaru grupy
4. Dodaje członków do grupy wybierając wierzchołki o największej liczbie sąsiadów
5. Powtarza proces dla wszystkich niepokrytych wierzchołków

Algorytm nie ma parametrów do dostrajania – wszystkie decyzje są podejmowane deterministycznie na podstawie struktury grafu i kosztów licencji.

Sortowanie według stopnia wierzchołka (liczby sąsiadów) sprawdza się dobrze w praktyce, ponieważ wierzchołki o wysokim stopniu mogą tworzyć większe, bardziej efektywne grupy licencyjne.

Algorithm 2 Algorytm zachłanny

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

posortuj wierzchołki malejąco według stopnia
$niepokryte \leftarrow V$
for każdy wierzchołek v w posortowanej kolejności **do**
    if v już pokryty **then continue**
    end if
    znajdź dostępnych sąsiadów v wśród niepokrytych
    for każdy typ licencji ℓ **do**
        oblicz efektywność: $koszt_\ell / rozmiar\_grupy$
    end for
    wybierz licencję i członków grupy o najlepszej efektywności
    utwórz grupę z v jako właścicielem
    usuń członków grupy z $niepokryte$
end for
return utworzone grupy

Złożoność i zastosowanie Algorytm ma złożoność czasową $O(nT + m \log n)$, gdzie n to liczba wierzchołków, T to liczba typów licencji, a m to liczba krawędzi. Algorytm zachłanny jest bardzo szybki i daje stabilne wyniki. Z tego powodu często używany był jako:

- podstawowa metoda do porównywania z innymi algorytmami,
- źródło rozwiązania początkowego dla bardziej zaawansowanych metod,
- szybka metoda dla dużych grafów, gdzie inne algorytmy są zbyt wolne.

Wadą algorytmu jest to, że podejmując lokalnie najlepsze decyzje, może przegapić lepsze rozwiązania globalne.

5.2.2 Heurystyka zbioru dominującego

Algorytm oparty na zbiorze dominującym korzysta z podejścia heurystycznego, szeroko opisywanego w literaturze w kontekście budowania maksymalnych zbiorów dominujących [9]. Na początku iteracyjnie wybierane są wierzchołki o najwyższej efektywności kosztowej, definiowanej jako stosunek liczby pokrywanych jeszcze węzłów do minimalnego kosztu licencji. Tak skonstruowany zbiór dominujący pokrywa cały graf. W naszej wersji algorytmu, rozszerzonej o obsługę różnych typów wierzchołków, po posortowaniu dominatorów według stopnia budowane są grupy obejmujące dominatora oraz jego nieprzydzielonych jeszcze sąsiadów. Każdej grupie przypisywana jest najtańsza licencja spełniająca ograniczenia pojemnościowe, a w przypadku braku takiej - licencja indywidualna.

Algorithm 3 Zbiór dominujący – heurystyka z przypisaniem grup

Require: graf $G = (V, E)$, $\mathcal{L}$

$U \leftarrow V$, $D \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
    dla każdego v policz coverage$(v) = |(N(v) \cup \{v\}) \cap U|$ i min_cpn(v)
    wybierz u maksymalizujące coverage(v)/ min_cpn(v); jeśli nie ma, weź dowolne $u \in U$
    $D \leftarrow D \cup \{u\}$, $U \leftarrow U \setminus (N(u) \cup \{u\})$
end while
posortuj D malejąco po deg
for każde $u \in D$ oraz dla pozostałych węzłów **do**
    $S \leftarrow (N(u) \cup \{u\}) \cap$ nieprzydzieleni
    wybierz najtańszą dopuszczalną grupę; w ostateczności licencję 1
end for
return grupy

Złożoność i zastosowanie Pierwsza faza (wybór dominatorów) w każdej rundzie przechodzi po wszystkich wierzchołkach, ich sąsiadach i typach licencji, co daje koszt rzędu $O(nmT)$. Druga faza, czyli budowanie grup, dla każdego dominatora sortuje jego sąsiadów i sprawdza wszystkie warianty licencji - w gęstych grafach rośnie to do $O(n^3 T \log n)$, a w rzadkich pozostaje bliżej $O(nmT)$. Wydajność zależy więc przede wszystkim od gęstości grafu i liczby dostępnych licencji.

5.2.3 Algorytm losowy

Algorytm losowy pełni rolę metody odniesienia, służącej do oceny jakości rozwiązań generowanych przez inne algorytmy. Jego zadaniem jest weryfikacja poprawności - jeżeli dana metoda uzyskuje wyniki gorsze niż losowy dobór licencji i grup, świadczy to o problemie w jej implementacji lub konfiguracji. Algorytm przetwarza wierzchołki w losowej kolejności i dla każdego z nich losowo wybiera typ licencji oraz skład grupy. W przypadkach, gdy losowy wybór nie jest możliwy (np. brak licencji spełniającej ograniczenia), stosowana jest prosta strategia zachłanna jako rozwiązanie awaryjne.

Idea metody Algorytm działa według następującej strategii:

1. Losuje kolejność przetwarzania wierzchołków.
2. Dla bieżącego wierzchołka wyznacza zbiór kandydatów obejmujący jego i nieprzydzielonych sąsiadów.
3. Jeżeli istnieje dopuszczalna licencja, losowo wybiera jej typ, rozmiar grupy oraz członków.
4. W przeciwnym razie przypisuje najtańszą dostępną licencję dopuszczalną dla danego węzła.
5. Proces powtarza się do pokrycia całego grafu; pozostałe nieprzydzielone węzły otrzymują indywidualne licencje.

Algorithm 4 Losowy – dobór licencji i składu grupy

Require: $G = (V, E)$, $\mathcal{L}$

$U \leftarrow V$, $\pi \leftarrow$ losowa permutacja V
for node w kolejności π **do**
    if $node \notin U$ **then continue**
    end if
    $S \leftarrow (N(node) \cup \{node\}) \cap U$
    if istnieje dopuszczalna licencja **then**
        wylosuj ℓ i rozmiar $s \in [l_\ell, \min\{|S|, u_\ell\}]$, dobierz losowych członków
    else
        wybierz najtańszą dopuszczalną grupę
    end if
    dodaj grupę, zaktualizuj U
end for
while $U \neq \emptyset$ **do** przypisz najtańszą licencję 1 i usuń węzeł z U
end while

Złożoność i zastosowanie Każdy wierzchołek i jego sąsiedzi są przeglądani co najwyżej raz, a dla każdej próby losowania licencji przechodzimy przez wszystkie typy licencji. Daje to koszt rzędu $O\big(T(m+n)\big)$, który w gęstych grafach upraszcza się do $O(Tn^2)$. Algorytm służy głównie jako punkt odniesienia: pozwala szybko ocenić, czy inne metody faktycznie generują lepsze rozwiązania niż czysty przypadek.

5.3 Metaheurystyki

Metaheurystyki to zaawansowane algorytmy przeszukujące przestrzeń rozwiązań w sposób inteligentny. W przeciwieństwie do heurystyk konstrukcyjnych, które budują rozwiązanie od zera, metaheurystyki zaczynają od pewnego rozwiązania i systematycznie je poprawiają.

Dobór parametrów Parametry metaheurystyk zostały dobrane eksperymentalnie na podstawie testów na grafach różnych rozmiarów.

Operacje modyfikacji rozwiązania Metaheurystyki poprawiają rozwiązanie stosując następujące operacje:

- zmiana typu licencji używanej przez właściciela grupy
- przeniesienie członka z jednej grupy do drugiej
- zamiana miejscami dwóch członków z różnych grup
- scal dwie grupy w jedną lub rozdziel na dwie dopuszczalne.

5.3.1 Algorytm genetyczny

Algorytm genetyczny utrzymuje populację pełnych przydziałów licencyjnych i z pokolenia na pokolenie ulepsza je, korzystając z losowych mutacji i krzyżowania par rodziców [28, 29]. Zaczyna od mieszanki rozwiązań zachłannych i losowych, a następnie w każdej generacji wybiera najlepsze osobniki (elita), losuje rodziców metodą turniejową i tworzy potomstwo przez krzyżowanie lub mutację. Słabsze rozwiązania są stopniowo zastępowane lepszymi, a algorytm zapamiętuje najlepszy znaleziony koszt.

Parametry

- **Wielkość populacji** $P = 30$ - liczba rozwiązań utrzymywanych w każdej generacji.
- **Liczba pokoleń** $G = 40$ - maksymalna liczba iteracji ewolucji.
- **Udział elity** $\alpha = 20\%$ - część najlepszych osobników kopiowana bez zmian do kolejnego pokolenia.
- **Prawdopodobieństwo krzyżowania** $p_c = 60\%$ - przy tej szansie dziecko powstaje przez połączenie dwóch rodziców; w przeciwnym razie wykonywana jest mutacja.

Algorithm 5 Algorytm genetyczny

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

```
utwórz populację początkową (zachłanny + losowe rozwiązania)
for każde pokolenie do
    oceń wszystkie rozwiązania (funkcja kosztu)
    zachowaj elitę (najlepsze rozwiązania)
    while populacja niepełna do
        if losowanie krzyżowania then
            wybierz dwóch rodziców (selekcja turniejowa)
            skrzyżuj rodziców (połącz efektywne grupy)
        else
            wybierz rozwiązanie i zmutuj (operacje sąsiedztwa)
        end if
        dodaj potomka do nowej populacji
    end while
    zaktualizuj najlepsze znalezione rozwiązanie
end for
return najlepsze rozwiązanie
```

Złożoność i zastosowanie Inicjalizacja populacji korzysta z jednej wersji zachłannej i P losowych rozwiązań, co kosztuje około $O\big(P \cdot (nT + m \log n)\big)$. Każda generacja sortuje populację ($O(P \log P)$), a następnie tworzy nowych potomków. Mutacje wywołują ograniczoną liczbę operatorów sąsiedztwa, a krzyżowanie w razie potrzeby uruchamia heurystykę zachłanną na podgrafie,

co razem daje koszt rzędu $O(nT + m \log n)$ na potomka. Łącznie otrzymujemy $O\big(G \cdot P \cdot (nT + m \log n)\big)$ w najgorszym przypadku. Algorytm działa wolniej od prostych heurystyk, ale potrafi znacząco poprawić ich wyniki i służy jako główna metoda poszukiwania wysokiej jakości rozwiązań, zwłaszcza gdy mamy czas na dłuższą optymalizację.

5.3.2 Przeszukiwanie tabu

Algorytm tabu startuje od rozwiązania zachłannego i w każdej iteracji generuje niewielkie sąsiedztwo kandydatów przy pomocy operatorów mutacji. Spośród poprawnych kandydatów wybierany jest najlepszy, o ile jego sygnatura nie znajduje się na liście tabu. Lista tabu blokuje cofanie ostatnich ruchów, a kryterium aspiracji pozwala ją złamać, gdy kandydat poprawia globalny wynik [30]. Dzięki temu algorytm balansuje między lokalnym doskonaleniem a eksploracją nowych przydziałów licencji.

Parametry

- **Maksymalna liczba iteracji** $I = 1000$ - liczba prób poprawienia rozwiązania.
- **Długość listy tabu** $L = 20$ - liczba ostatnich ruchów blokowanych przed ponownym użyciem.
- **Liczba sąsiadów na iterację** $k = 10$ - liczba kandydatów generowanych przed wyborem najlepszego.

Algorithm 6 Przeszukiwanie tabu

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

$aktualne \leftarrow$ rozwiązanie początkowe (zachłanne)
$najlepsze \leftarrow aktualne$
$lista\_tabu \leftarrow$ pusta lista o stałej długości
for każdą iterację **do**
    wygeneruj sąsiadów $aktualne$ (zmiana licencji, przeniesienie członka, itp.)
    wybierz najlepszego sąsiada spośród tych, które nie są na liście tabu lub poprawiają dotychczasowe $najlepsze$ (aspiracja)
    if znaleziono kandydata **then**
        $aktualne \leftarrow$ wybrany kandydat
        dodaj ruch do $lista\_tabu$
        if $aktualne$ lepsze niż $najlepsze$ **then**
            $najlepsze \leftarrow aktualne$
        end if
    end if
end for
return $najlepsze$

Złożoność i zastosowanie Rozruch algorytmu obejmuje pojedyncze wywołanie heurystyki zachłannej $O(nT + m \log n)$. W każdej z I iteracji generowanych jest do k sąsiadów, a każdy kandydat przechodzi walidację obejmującą sprawdzenie pojemności, pokrycia i sąsiedztwa, co kosztuje około $O(nT + m)$. Całość prowadzi do kosztu rzędu $O\big(I \cdot k \cdot (nT + m)\big)$. Przeszukiwanie tabu dobrze sprawdza się jako metoda doszkalająca: potrafi poprawić rozwiązania bazowe przy umiarkowanym czasie obliczeń i jest odporne na zbieganie w krótkie cykle dzięki liście tabu.

5.3.3 Algorytm mrówkowy

Algorytm mrówkowy buduje wiele rozwiązań równolegle. Każda mrówka konstruuje przydział licencji, kierując się siłą śladów feromonowych (informacja o dotychczas dobrych wyborach) oraz heurystyką preferującą wierzchołki o dużym stopniu i licencje o dobrym stosunku pojemności do ceny [31]. Po każdej iteracji feromony parują, a najlepsze dotąd rozwiązanie wzmacnia ścieżki, dzięki czemu kolejne mrówki chętniej eksplorują obiecujące fragmenty przestrzeni.

Parametry

- **Waga feromonu** $\alpha = 1.0$ – określa, jak mocno mrówki ufają dotychczasowym śladom.
- **Waga heurystyki** $\beta = 2.0$ – wzmacnia preferencję dla lokalnie dobrych decyzji (wysoki stopień, tanie licencje).
- **Tempo parowania** $\rho = 0.5$ – część feromonu usuwana po każdej iteracji.
- **Prawdopodobieństwo wyboru zachłannego** $q_0 = 0.9$ – z tą szansą mrówka wybiera najlepszą dostępnie opcję, w przeciwnym razie losuje proporcjonalnie do wag.
- **Liczba mrówek** $A = 20$ – ile rozwiązań konstruujemy równolegle w jednej iteracji.
- **Maksymalna liczba iteracji** $I = 100$ – ile razy aktualizujemy feromony.
- **Losowe nasiono** (opcjonalne) – pozwala odtworzyć przebieg eksperymentu.

Algorithm 7 Algorytm mrówkowy

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

zainicjalizuj feromony τ (dla par wierzchołek-licencja)
zainicjalizuj heurystyki η (na podstawie stopni wierzchołków i kosztów licencji)
$najlepsze \leftarrow$ rozwiązanie początkowe (zachłanne)
for każdą iterację **do**
    for każdą mrówkę **do**
        $niepokryte \leftarrow V$
        while $niepokryte \neq \emptyset$ **do**
            wybierz właściciela na podstawie τ i η (reguła wyboru lub ruletka)
            wybierz typ licencji na podstawie τ i η
            utwórz grupę, usuń członków z $niepokryte$
        end while
        if mrówka znalazła lepsze rozwiązanie **then**
            $najlepsze \leftarrow$ rozwiązanie mrówki
        end if
    end for
    wyparuj część feromonów: $\tau \leftarrow \tau \cdot (1 - evaporation)$
    wzmocnij feromony na ścieżce $najlepsze$: $\tau \leftarrow \tau + 1/koszt$
end for
return $najlepsze$

Złożoność i zastosowanie Inicjalizacja feromonów i heurystyk kosztuje $O(nT)$. Pojedyncza konstrukcja rozwiązania przez mrówkę odwiedza każdy wierzchołek co najwyżej raz, ale przy wyborze właściciela ponownie ocenia wszystkich kandydatów i wszystkie licencje; prowadzi to do kosztu około $O(n^2T + m \log n)$ na jedną mrówkę (uwzględniając sortowanie sąsiadów). Cały algorytm wykonujący I iteracji z A mrówkami ma więc koszt rzędu $O\big(I \cdot A \cdot (n^2T + m \log n)\big)$. Metoda jest cięższa obliczeniowo niż tabu czy zachłanny, ale dobrze sprawdza się, gdy chcemy eksplorować wiele alternatywnych konfiguracji i stopniowo wzmacniać najlepsze z nich.

5.3.4 Symulowane wyżarzanie

Symulowane wyżarzanie rozpoczyna od rozwiązania zachłannego i w każdej iteracji losuje ruch sąsiedztwa (zmiana licencji, przeniesienie członka, zamiana, scalenie lub podział grupy). Nowy stan jest akceptowany zawsze, gdy obniża koszt, a czasami także wtedy, gdy go pogarsza - z prawdopodobieństwem zależnym od bieżącej temperatury T i różnicy kosztów [32]. Temperatura maleje według ustalonego współczynnika, a gdy algorytm zbyt długo nie znajduje poprawy, dodatkowo jest obniżana o połowę. Dzięki temu metoda potrafi opuszczać lokalne minima i stopniowo stabilizuje się w pobliżu dobrego rozwiązania.

Parametry

- **Temperatura początkowa** $T_0 = 100.0$ - początkowy poziom losowości przy akceptacji gorszych ruchów.
- **Współczynnik chłodzenia** $\alpha = 0.995$ - w każdej iteracji temperatura jest mnożona przez α.
- **Temperatura minimalna** $T_{\text{min}} = 0.001$ - po jej osiągnięciu algorytm kończy działanie.
- **Maksymalna liczba iteracji** $I = 20\,000$ - górne ograniczenie liczby kroków.
- **Limit stagnacji** $S = 2\,000$ - po tylu nieudanych próbach temperatura jest dodatkowo dzielona przez 2.

Algorithm 8 Symulowane wyżarzanie

Require: graf $G = (V, E)$, typy licencji $\mathcal{L}$

$aktualne \leftarrow$ rozwiązanie początkowe (zachłanne)
$najlepsze \leftarrow aktualne$
$temperatura \leftarrow T_0$ (początkowa temperatura)
for każdą iterację **do**
    if $temperatura < T_{min}$ **then break**
    end if
    wybierz losową operację sąsiedztwa
    $kandydat \leftarrow$ wynik operacji sąsiedztwa
    $\Delta \leftarrow koszt(kandydat) - koszt(aktualne)$
    if $\Delta \leq 0$ LUB $random() < \exp(-\Delta / temperatura)$ **then**
        $aktualne \leftarrow kandydat$
        if $koszt(kandydat) < koszt(najlepsze)$ **then**
            $najlepsze \leftarrow kandydat$
        end if
    end if
    $temperatura \leftarrow temperatura \cdot cooling\_rate$
end for
return $najlepsze$

Złożoność i zastosowanie Inicjalizacja wymaga jednego wywołania heurystyki zachłannej $O(nT + m \log n)$. Każda iteracja wykonuje ograniczoną liczbę prób wygenerowania sąsiada (do 12 ruchów), a zaakceptowany kandydat przechodzi pełną walidację pokrycia i ograniczeń, co kosztuje około $O(nT + m)$. W rezultacie złożoność całego przebiegu wynosi $O\big(I \cdot (nT + m)\big)$ z dodatkowym mnożnikiem wynikającym z limitu stagnacji. Symulowane wyżarzanie jest umiarkowanie kosztowne, ale często zapewnia lepsze rozwiązania niż czyste metody lokalne, szczególnie gdy potrzebna jest możliwość wychodzenia z lokalnych minimów przy ograniczonym czasie działania.

6. EKSPERYMENTY DLA LICENCJI DUOLINGO SUPER

6.1 Środowisko i metodologia

Eksperymenty wykonano na komputerze z procesorem Intel Core i5-14600KF (12 rdzeni ograniczone przez konfigurację WSL2, taktowanie 3,49 GHz) oraz 16 GB pamięci RAM. System operacyjny stanowił Ubuntu 24.04 LTS uruchomiony w środowisku WSL2. Implementacja algorytmów powstała w języku Python 3.13; wykorzystano biblioteki NumPy, NetworkX, a także PuLP jako interfejs do solverów ILP. Każdy pomiar obejmował wyłącznie czas obliczeń, pomijając koszt przygotowania instancji oraz operacje wejścia/wyjścia.

Analiza objęła dwa zestawy danych: syntetyczne grafy losowe, bezskalowe oraz małoświatowe o liczebności od 50 do 450 wierzchołków, a także grafy ego z serwisu Facebook o liczebności od 53 do 1035 wierzchołków. Dla każdej instancji uruchamiano wszystkie algorytmy z limitem czasu wynoszącym 60 sekund, a przekroczenie tego limitu oznaczano jako timeout i wyłączano z dalszych obliczeń. Na grafach syntetycznych odnotowano 10 timeoutów dla algorytmu mrówkowego oraz solvera ILP, natomiast na grafach rzeczywistych odpowiednio 8, 18 i 6 przypadków dla algorytmu mrówkowego, solvera ILP oraz przeszukiwania tabu. Wszystkie pozostałe uruchomienia zakończyły się sukcesem.

W przypadku grafów syntetycznych dla każdego rozmiaru generowano trzy niezależne instancje, a następnie wykonywano po dwa uruchomienia algorytmów na każdej z nich. Analiza sieci ego Facebooka obejmowała po jednym grafie na rozmiar oraz dwa powtórzenia dla każdej pary (graf, algorytm), co pozwoliło oszacować zmienność wyników przy zachowaniu rozsądnego budżetu obliczeniowego. W każdej próbie zapisano całkowity koszt licencji, czas wykonania oraz koszt w przeliczeniu na wierzchołek. Wyniki agregowano przez średnią. Spójność różnic między algorytmami oceniono przy użyciu testów Friedmana, gdzie χ_F^2 to wartość statystyki testu Friedmana oraz p to poziom istotności, a następnie następujących po nich porównań post-hoc metodą Nemenyi'ego. Na syntetycznym benchmarku otrzymano statystyki $\chi_F^2 = 577{,}93$ dla czasu oraz $\chi_F^2 = 518{,}01$ dla kosztu na węzeł ($p < 10^{-100}$ w obu przypadkach), co uzasadnia szczegółowe porównania par algorytmów.

Dodatkowo, w celu ułatwienia porównań między różnymi typami licencji, ceny każdej z licencji zostały znormalizowane. Cena licencji indywidualnej została ustalona na poziomie 1, a ceny licencji grupowych przeskalowano zgodnie z ich pierwotnym stosunkiem do ceny licencji indywidualnej. Dzięki temu możliwe było bardziej przejrzyste porównanie kosztów między różnymi strategiami licencjonowania, niezależnie od ich bezwzględnych wartości.

6.2 *Duolingo Super na grafach syntetycznych*

6.2.1 Statystyki zbiorcze

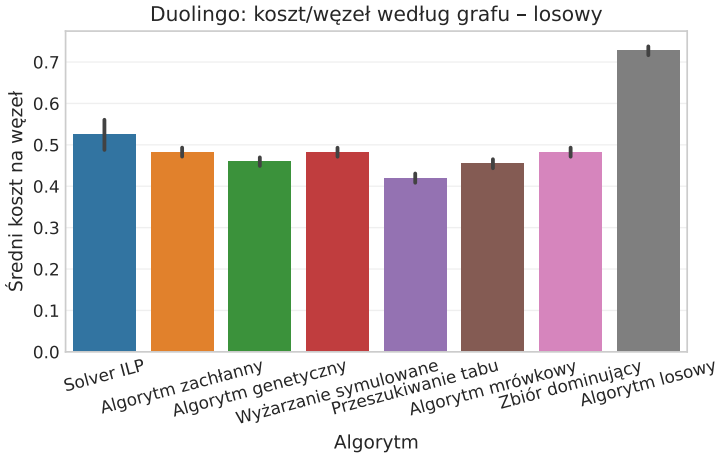
Tabela 6.1 zestawia średnie wartości kosztu na węzeł oraz czasu dla licencji Duolingo Super. **Solver ILP** pozostaje najlepszym punktem odniesienia jakościowego (średni koszt na węzeł 0,445), lecz przydaje się jedynie dla mniejszych instancji. Posiada tyle samo timeoutów co **Algorytm mrówkowy**. Wśród metaheurystyk najniższy koszt na węzeł osiąga właśnie **Algorytm mrówkowy** (0,506), natomiast **Przeszukiwanie tabu** zapewnia najlepszy kompromis kosztu na węzeł i czasu w grupie metod przybliżonych. **Algorytm zachłanny** i **Algorytm losowy** działają prawie natychmiast, ale tylko pierwszy z nich zachowuje akceptowalny koszt na węzeł (0,548), podczas gdy losowy baseline pozostaje wyraźnie gorszy jakościowo.

Tabela 6.1: Średnie wartości kosztu na węzeł i czasu dla licencji Duolingo Super na grafach syntetycznych.

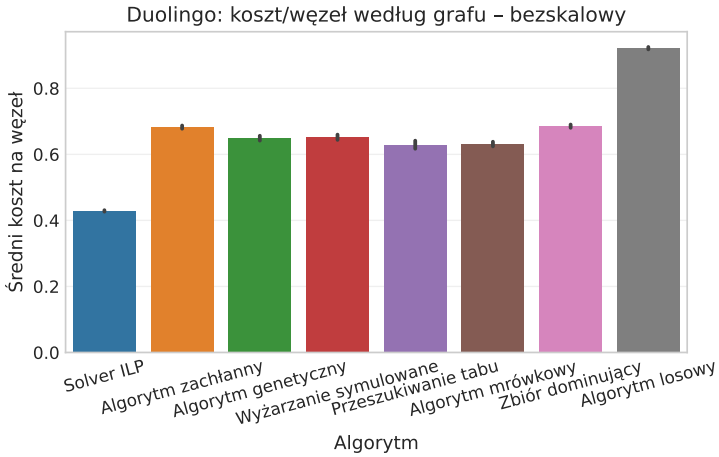
Algorytm	Średni koszt/węzeł	Średni koszt	Średni czas [s]
Solver ILP	0.445	73.83	2.492
Algorytm mrówkowy	0.506	83.58	5.124
Przeszukiwanie tabu	0.500	117.80	3.298
Algorytm genetyczny	0.515	118.36	1.222
Wyżarzanie symulowane	0.531	120.90	0.975
Zbiór dominujący	0.542	119.97	0.017
Algorytm zachłanny	0.548	121.94	0.001
Algorytm losowy	0.799	179.90	0.001

6.2.2 Porównanie algorytmów na grafach syntetycznych

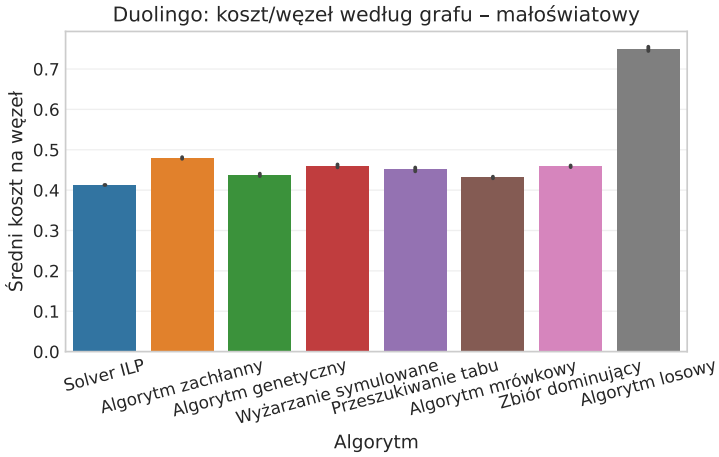
Rysunki 6.1–6.3 pokazują, że algorytmy dla licencji Duolingo Super osiągają wyniki porównywalne z solverem ILP w przypadku grafów losowych i małoświatowych. Natomiast w przypadku grafów bezskalowych wyniki są zauważalnie gorsze.



Rysunek 6.1: Koszt na węzeł w zależności od struktury grafu losowej.

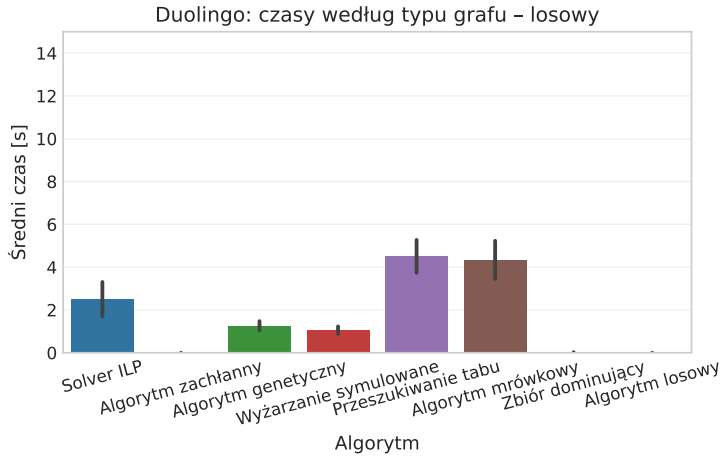


Rysunek 6.2: Koszt na węzeł w zależności od struktury grafu bezskalowej.

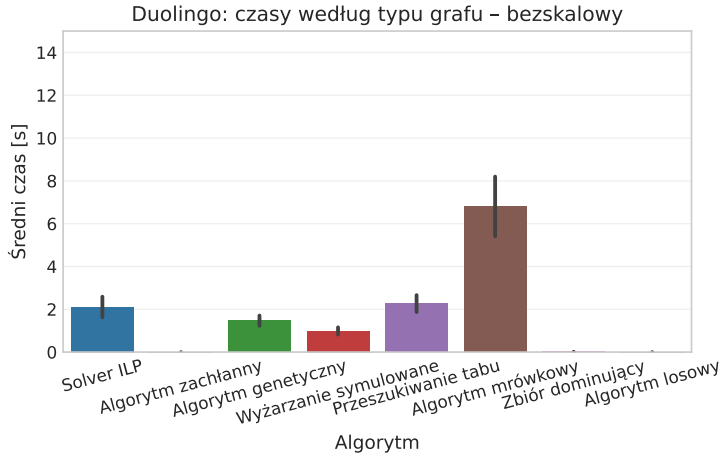


Rysunek 6.3: Koszt na węzeł w zależności od struktury grafu małoświatowej.

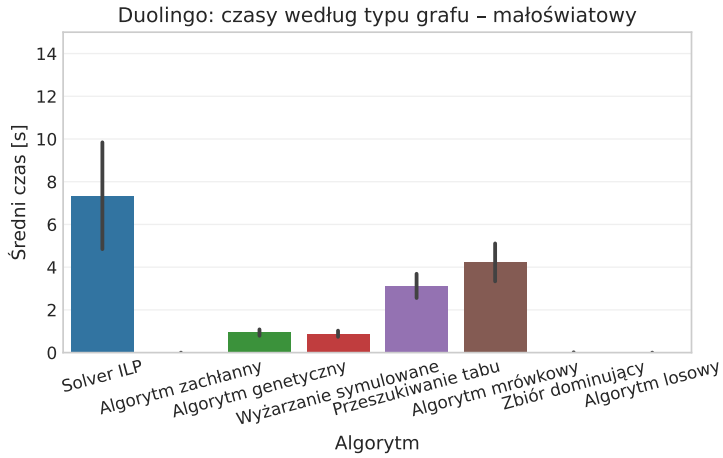
Podobne obserwacje dotyczą czasów wykonania, jak ilustrują rysunki 6.4–6.6. Czasy działania algorytmów są zbliżone dla różnych typów grafów, z wyjątkiem solvera ILP, który średnio działa szybciej na grafach bezskalowych, oraz przeszukiwania tabu, które działa na nich dłużej. Pozostałe algorytmy wykazują porównywalne czasy działania niezależnie od struktury grafu. Tabela 6.2 podsumowuje średnie czasów i kosztów na węzeł dla różnych typów grafów.



Rysunek 6.4: Czas wykonania w zależności od struktury grafu losowej.



Rysunek 6.5: Czas wykonania w zależności od struktury grafu bezskalowej.



Rysunek 6.6: Czas wykonania w zależności od struktury grafu małoświatowej.

Tabela 6.2: Średnie koszty i czasy na węzeł dla różnych typów grafów (Duolingo Super i dominowanie rzymskie).

Licencja	Typ grafu	Śr. koszt/węzeł	Śr. czas [s]
Duolingo Super	Bezskalowy	0.661	1.655
Duolingo Super	Losowy	0.502	1.598
Duolingo Super	Małoświatowy	0.495	1.346
Dominowanie rzymskie	Bezskalowy	0.490	0.913
Dominowanie rzymskie	Losowy	0.344	0.815
Dominowanie rzymskie	Małoświatowy	0.409	1.544

Z powyższych danych wynika, że struktura grafu ma istotny wpływ zarówno na koszty, jak i na czasy działania algorytmów. Licencja Duolingo Super radzi sobie najgorzej w przypadku grafów bezskalowych. Z kolei licencja dominowania rzymskiego osiąga najgorsze wyniki dla grafów małoświatowych, podczas gdy w przypadku grafów bezskalowych wypada najlepiej.

6.3 Duolingo Super na grafach rzeczywistych

W analizie grafów ego z serwisu Facebook pominięto obserwacje, w których solver ILP nie zakończył pracy przed limitem czasu (18 przypadków). Dodatkowo odnotowano 8 timeoutów algorytmu mrówkowego i 6 przypadków w przeszukiwaniu tabu; pozostałe uruchomienia zakończyły się sukcesem.

Tabela 6.3: Statystyki kosztu i czasu dla licencji Duolingo Super na grafach rzeczywistych.

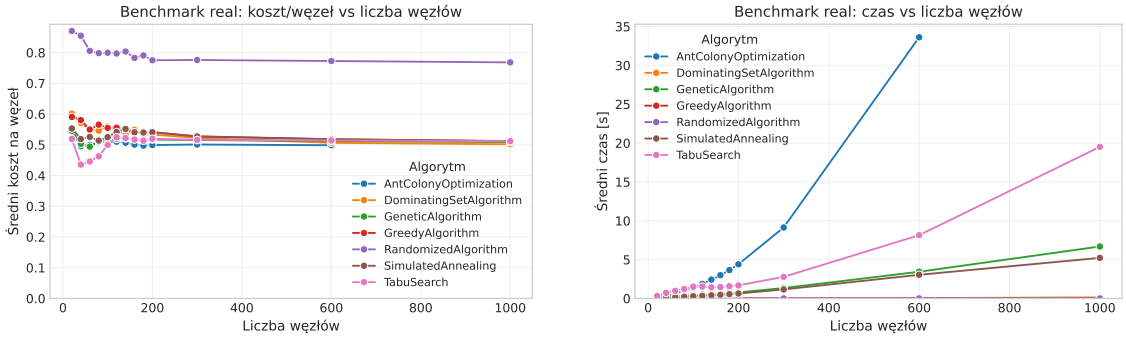
Algorytm	Średni koszt	Śr. koszt/węzeł	Śr. czas [s]
Algorytm mrówkowy	83.58	0.506	5.124
Zbiór dominujący	119.97	0.542	0.017
Algorytm genetyczny	118.36	0.515	1.222
Algorytm zachłanny	121.94	0.548	0.001
Algorytm losowy	179.90	0.799	0.001
Wyżarzanie symulowane	120.90	0.531	0.975
Przeszukiwanie tabu	117.80	0.500	3.298

Tabela 6.3 pokazuje, że zarówno przeszukiwanie tabu, jak i algorytm mrówkowy zachowują przewagę kosztową nad heurystykami losowymi i zachłannymi, choć okupują ją dłuższym czasem działania.

6.3.1 Skalowanie i jakość

Rysunek 6.7 pokazuje, że wraz ze wzrostem liczby wierzchołków koszt na węzeł rośnie umiarkowanie, przy czym przeszukiwanie tabu utrzymuje najniższe wartości, a algorytm mrówkowy plasuje się tuż za nim. Czasy działania wszystkich metod mieszczą się w przedziale do

kilku sekund i rosną łagodnie wraz z rozmiarem grafu; algorytm zachłanny nadal pozostaje niemal natychmiastowy i stanowi dobry punkt startowy dla metaheurystyk. Dodatkowo tabela 6.4 zbiera średnie wartości (TabuSearch) dla kolejnych rozmiarów sieci ego i pokazuje, że koszt na węzeł utrzymuje się w przedziale 1,8–2,5 przy czasie rosnącym od ułamka sekundy do około 5,5 s dla największych grafów.



Rysunek 6.7: Koszt na węzeł i czas wykonania licencji Duolingo Super w funkcji liczby wierzchołków (grafy ego Facebook).

Tabela 6.4: Średni koszt na węzeł i czas (przeszukiwanie tabu) względem liczby wierzchołków w sieciach ego Facebook.

Liczba wierzchołków	Śr. koszt/węzeł	Śr. czas [ms]
53	6.175	2161.349
62	5.175	3141.926
151	5.297	6825.722
169	5.499	8726.145
225	5.704	12324.041
334	7.843	32705.396
535	7.400	58557.464

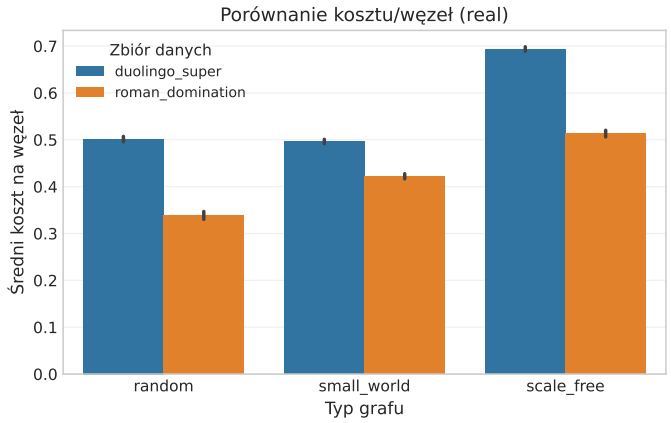
6.4 Porównanie z dominowaniem rzymskim

Porównania z dominowaniem rzymskim ograniczono do wspólnych instancji i algorytmów. Rysunki 6.8–6.10 zestawiają różnice w kosztach, czasach i strukturze licencji, a tabela 6.5 gromadzi średnie według typu grafu syntetycznego.

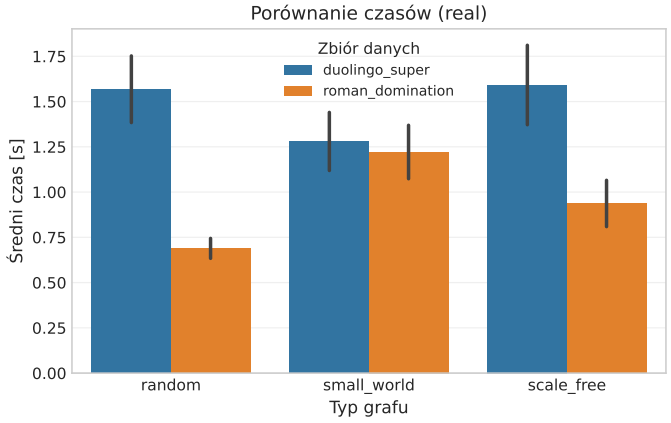
Tabela 6.5: Średnie czasu i kosztu na węzeł według typu grafu (wspólne instancje).

Typ grafu	Duolingo Super		Dominowanie rzymskie	
	Czas [s]	Koszt/węzeł	Czas [s]	Koszt/węzeł
Losowy	1.598	0.502	0.815	0.344
Bezskalowy	1.655	0.661	0.913	0.490
Małoświatowy	1.346	0.495	1.544	0.409

Dominowanie rzymskie zapewnia niższy koszt na węzeł we wszystkich porównywanych rodzinach grafów. Największa różnica dotyczy struktur losowych, gdzie przewaga wynosi ok. 0,16 punktu (32%). W sieciach bezskalowych różnica maleje do 0,17, natomiast w małoświatowych do 0,09. Czasy wykonania są porównywalne, przy lekkiej przewadze Duolingo Super w grafach małoświatowych, ale dominowanie rzymskie jest szybsze w grafach losowych i bezskalowych. Rysunek 6.8 ilustruje te obserwacje dla pełnego rozkładu, a rysunek 6.9 prezentuje analogiczne dane czasowe. Warto zauważyć, że wyższe koszty na węzeł w przypadku Duolingo Super wynikają z ograniczeń w opłacalności licencji grupowych. Licencja grupowa jest około 2,1 razy droższa od indywidualnej, co oznacza, że opłaca się ją tworzyć dopiero dla grup liczących co najmniej trzy osoby (właściciel i dwóch sąsiadów). W efekcie liczba licencji indywidualnych w Duolingo Super jest większa, co przekłada się na wyższy średni koszt na węzeł w porównaniu do dominowania rzymskiego.



Rysunek 6.8: Koszt na węzeł według typu grafu: porównanie licencji Duolingo Super i dominowania rzymskiego.

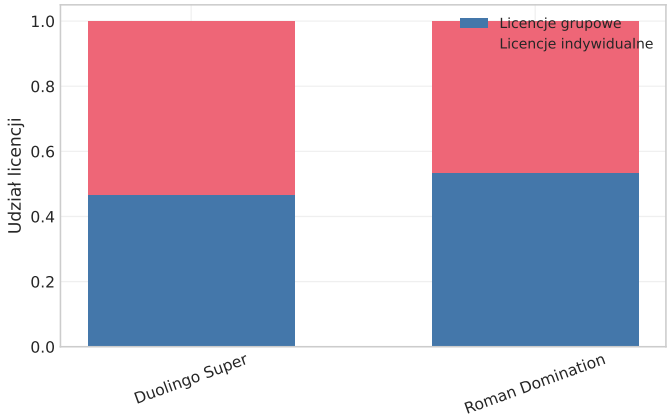


Rysunek 6.9: Czas wykonania według typu grafu: porównanie licencji Duolingo Super i dominowania rzymskiego.

Rysunek 6.10 pokazuje, że dominowanie rzymskie charakteryzuje się wyższym stosunkiem licencji grupowych do indywidualnych w porównaniu do Duolingo Super. W przypadku dominowania rzymskiego stosunek ten wynosi około 1,15:1, podczas gdy dla Duolingo Super jest to

około 0,88:1.

Różnica ta wynika z faktu, że dominowanie rzymskie nie posiada ograniczenia maksymalnej pojemności grupy licencyjnej. W przypadku Duolingo Super, gdy grupa osiąga maksymalną pojemność, dodatkowi użytkownicy, którzy mogliby do niej należeć, są zmuszeni do zakupu licencji indywidualnych. W pełnym zbiorze danych, obejmującym trzy typy grafów syntetycznych, zaobserwowano 137690 licencji w Duolingo Super, z czego 73176 (53,15%) to licencje indywidualne. W przypadku dominowania rzymskiego, dzięki braku ograniczeń pojemności grup, liczba licencji indywidualnych jest proporcjonalnie niższa.



Rysunek 6.10: Struktura wykorzystania licencji: porównanie licencji Duolingo Super i dominowania rzymskiego.

6.5 Wnioski

Analiza potwierdza dużą stabilność wyników Duolingo Super: metaheurystyki utrzymują przewagę jakości nad prostymi heurystykami niezależnie od rozmiaru i struktury grafu, a **Algorytm mrówkowy** oraz **Przeszukiwanie tabu** stanowią najbardziej konkurencyjną parę. Równocześnie dominowanie rzymskie pozostaje trudne do pobicia pod względem kosztu na węzeł, głównie dzięki większemu wykorzystaniu licencji grupowych i niższemu mnożnikowi kosztów jednostkowych.

7. SYMULACJA DYNAMICZNA

7.1 Założenia i konfiguracja

Eksperymenty dynamiczne wykonano w tym samym środowisku obliczeniowym co testy statyczne. Każda symulacja obejmuje 30 kroków, a po każdej mutacji sieci algorytmy otrzymują maksymalnie 45 s na ponowne zbilansowanie licencji. Analizowano dwie konfiguracje licencyjne – Duolingo Super oraz dominowanie rzymskie. Do testów wykorzystano różne typy grafów: losowe, bezskalowe, małoświatowe oraz grafy bez ego Facebook.

Testy przeprowadzono dla sześciu profili mutacji: trzech syntetycznych, które charakteryzują się prostą logiką modyfikacji (low, med, high), oraz trzech bardziej złożonych i realistycznych, które uwzględniają mechanizmy takie jak preferencyjne przyłączanie, domykanie triad czy losowe przekształcanie krawędzi (pref_triadic, pref_pref, rand_rewire).

Warianty *low*, *med* oraz *high* różnią się prawdopodobieństwami modyfikacji wierzchołków i krawędzi (tabela 7.1).

Tabela 7.1: Parametry intensywności mutacji w symulacji dynamicznej.

Poziom	Dodawanie węzłów	Usuwanie węzłów	Dodawanie krawędzi	Usuwanie krawędzi
low	0.02	0.01	0.06	0.04
mid	0.06	0.04	0.18	0.12
high	0.12	0.08	0.30	0.20

Dodatkowo zbadano trzy bardziej realistyczne profile ewolucji sieci. Wszystkie operują na tych samych limitach liczby dodawanych/usuwanych elementów co warianty syntetyczne, różnią się jednak mechanizmem wyboru sąsiedztwa.

Wariant *pref_triadic* łączy dwa mechanizmy typowe dla sieci społecznych: preferencyjne przyłączanie i domykanie triad. Nowe węzły dołączają do sieci, preferencyjnie łącząc się z istniejącymi węzłami o wysokim stopniu (zasada ”bogaci stają się bogatsi”). Nowe krawędzie powstają natomiast poprzez domykanie trójkątów – wyszukiwanie dwóch niepołączonych węzłów, które mają wspólnego sąsiada, i tworzenie między nimi połączenia. Mechanizm ten wzmacnia lokalną strukturę klastrową sieci, co odzwierciedla tendencję do tworzenia zamkniętych grup w rzeczywistych sieciach.

Wariant *pref_pref* stosuje mechanizm preferencyjnego przyłączania zarówno przy dodawaniu nowych węzłów, jak i tworzeniu krawędzi między już istniejącymi. Oznacza to, że nie tylko nowe węzły chętniej łączą się z popularnymi wierzchołkami, ale również nowe krawędzie z większym prawdopodobieństwem powstaną między dwoma węzłami, które już teraz mają wysoki stopień. Takie podejście intensywnie promuje tworzenie centralnych hubów w sieci, co jest kluczową cechą topologii bezskalowych.

Wreszcie wariant *rand_rewire* wprowadza do sieci element losowości, inspirowany modelem Wattsa–Strogatza. Nowe węzły dołączane są w sposób losowy, bez preferencji dla popularniejszych wierzchołków. Zamiast dodawania nowych krawędzi, model ten modyfikuje istniejącą strukturę poprzez operację ”przełączania” (rewiring): losowa krawędź jest usuwana, a jeden z jej końców jest następnie łączony z innym, losowo wybranym węzłem w sieci. Proces ten prowadzi do powstawania ”skrótów” i zmniejsza regularność struktury grafu.

7.2 Algorytm zachłanny

W tym wariancie algorytm zachłanny w każdym kroku symulacji buduje rozwiązanie całkowicie od zera. Stanowi to dolną granicę narzutu czasowego dla metod bez pamięci oraz bazowy punkt odniesienia jakości, do którego porównujemy bardziej zaawansowane metaheurystyki korzystające z rozwiązań z poprzedniego kroku.

7.2.1 Zestawienie dla wszystkich mutacji

Tabela 7.2 zestawia średni koszt na węzeł i średni czas na krok dla sześciu badanych profili mutacji: trzech syntetycznych oraz trzech realistycznych. Wszystkie czasy mieszczą się w przedziale 0.5–1.8 ms na krok, a średni koszt na węzeł oscyluje wokół 0.46–0.48.

Tabela 7.2: Algorytm zachłanny: średni koszt na węzeł oraz średni czas na krok dla wszystkich wariantów mutacji.

Metoda mutacji	Koszt/węzeł (mean)	Średni czas [s]
high	0.48009	0.00159
low	0.47983	0.00165
med	0.47491	0.00181
pref_pref	0.46371	0.00083
pref_triadic	0.46787	0.00053
rand_rewire	0.47556	0.00083

Algorytm zachłanny jest bardzo szybki, co potwierdza jego przydatność jako lekki baseline czasowy w środowisku dynamicznym. Warianty realistyczne sprzyjają nieco niższym kosztom: `pref_pref` osiąga najniższy średni koszt (0.464), a `pref_triadic` jest jednocześnie najszybszy ($0.000\,53$ s). Wariant `rand_rewire` jest trudniejszy (0.476), ale pozostaje bardzo szybki czasowo.

Różnice kosztu dla mutacji syntetycznych są niewielkie (0.475–0.480), natomiast czasy są wyższe niż w profilach realistycznych, szczególnie dla `med/high`. Wskazuje to, że bardziej lokalne, realistyczne przekształcenia struktury grafu są łatwiejsze do obsłużenia. Czasy dla `pref_pref` wynoszą średnio $0.000\,83$ s, a dla `pref_triadic` $0.000\,53$ s, co potwierdza ich przewagę czasową nad mutacjami syntetycznymi ($0.001\,59$ s dla `high`).

7.3 Wyniki na mutacjach syntetycznych

7.3.1 Metaheurystyki

Tabela 7.3 przedstawia zbiorcze wyniki dla różnych metod mutacji. Średni koszt na węzeł jest bardzo zbliżony dla wszystkich poziomów intensywności, z jedynie nieznacznym wzrostem dla wariantu `high`. Sugeruje to, że algorytmy są w stanie skutecznie adaptować się do zmian w topologii sieci.

Średni czas wykonania rośnie wraz z intensywnością mutacji. Wariant `low` jest najszybszy, podczas gdy `high` wymaga najwięcej czasu na ponowne zbilansowanie. Jest to naturalna konsekwencja faktu, że większa liczba modyfikacji grafu (dodawanie/usuwanie węzłów i krawędzi) stanowi większe wyzwanie obliczeniowe dla algorytmów optymalizacyjnych. Mimo to, różnice w czasach nie są drastyczne, co świadczy o dobrej skalowalności zastosowanych metod.

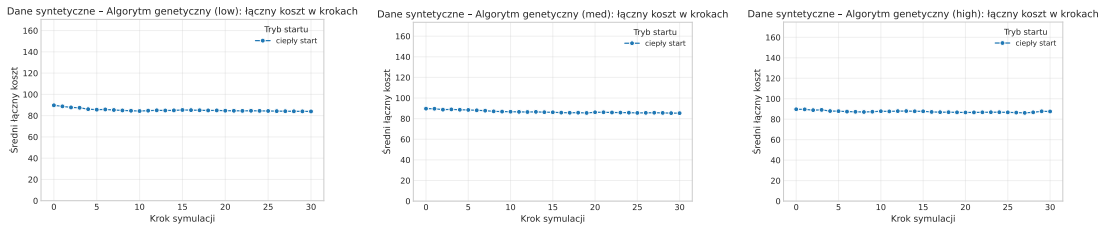
Tabela 7.3: Wyniki dla różnych metod mutacji.

Metoda mutacji	**Średni koszt**	**Średni czas [s]**
high	0.4893	3.169
med	0.4850	3.075
low	0.4852	2.878

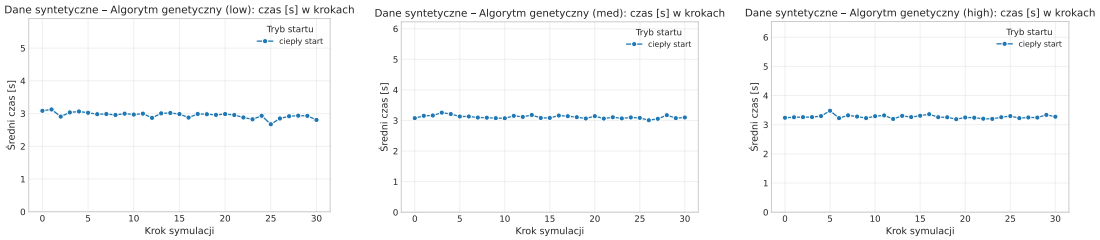
7.3.2 Profil kosztu i czasu w czasie

Jako przykład ewolucji kosztu i czasu w krokach symulacji wybrano algorytm genetyczny. Rysunki 7.1 i 7.2 przedstawiają odpowiednio przebieg kosztu na węzeł oraz czasu wykonania w zależności od kroku symulacji dla różnych poziomów intensywności mutacji.

Dla wariantu `high` można zaobserwować niewielkie, lecz zauważalne wahania średniego kosztu na węzeł, które nie są tak widoczne w przypadku wariantów `low` i `med`. Jeśli chodzi o czasy wykonania, dla każdego z trzech wariantów widoczne są zbliżone wahania w trakcie trwania symulacji.



Rysunek 7.1: Algorytm genetyczny – koszt na węzeł w funkcji kroku (warianty low/med/high).



Rysunek 7.2: Algorytm genetyczny – czas wykonania w funkcji kroku (warianty low/med/high).

7.4 Wyniki na mutacjach realistycznych

Tabela 7.4 przedstawia zbiorcze wyniki dla scenariuszy realistycznych. Wariant `pref_triadic` wyróżnia się najkrótszym średnim czasem wykonania (poniżej 1 sekundy), przy zachowaniu kosztu na poziomie zbliżonym do wariantu `pref_pref`. Scenariusz `rand_rewire` okazał się najtrudniejszy – charakteryzuje się zarówno najwyższym średnim kosztem, jak i najdłuższym czasem przetwarzania.

Tabela 7.4: Wyniki dla różnych metod mutacji w scenariuszach realistycznych.

Metoda mutacji	**Średni koszt całkowity**	**Koszt/węzeł (mean)**	**Średni czas [s]**
pref_pref	113.34	0.4764	1.6774
pref_triadic	70.08	0.4764	0.8619
rand_rewire	116.83	0.4917	1.8421

7.4.1 Wybrane algorytmy i metody mutacji

Poniżej wybrano kilka par algorytmów i metod mutacji, które pokazują różne kompromisy. Dla porównania uwzględniono również dwie mutacje syntetyczne. Nie są to zawsze najlepsze jakościowo konfiguracje, ale dobrze ilustrują różne scenariusze.

Tabela 7.5: Wybrane pary algorytmów i metod mutacji (różne kompromisy).

Algorytm	Metoda	Liczba kroków	Koszt/węzeł	Śr. czas [s]
Solver ILP	pref_triadic	434	0.362	1.525
Solver ILP	rand_rewire	496	0.390	2.553
Algorytm genetyczny	pref_triadic	744	0.409	0.615
Algorytm genetyczny	pref_pref	930	0.414	1.134
Przeszukiwanie tabu	pref_triadic	744	0.413	1.470
Przeszukiwanie tabu	rand_rewire	930	0.445	2.581
Algorytm mrówkowy	pref_pref	899	0.417	6.906
Algorytm mrówkowy	pref_triadic	744	0.424	3.002
Wyżarzanie symulowane	pref_triadic	744	0.460	0.555
Algorytm zachłanny	pref_pref	930	0.464	0.001
Zbiór dominujący	pref_triadic	744	0.457	0.005
Algorytm losowy	pref_pref	930	0.754	0.001

Solver ILP osiąga najniższe koszty, ale wymaga około 1.5–2.6 sekundy na krok. Algorytm genetyczny dobrze sprawdza się przy zmianach klastrowych (`pref_triadic`), oferując niski koszt i czas około 0.6 sekundy. Przy wariancie `pref_pref` jest wolniejszy (1.1 s), ale zachowuje dobrą jakość. Z kolei wyżarzanie symulowane jest szybkie (około pół sekundy) i stabilne, ale jakościowo ustępuje bardziej zaawansowanym metodom.

Przeszukiwanie tabu korzysta z lokalności zmian, osiągając sensowny koszt i czas około 1.5 sekundy przy `pref_triadic`. Jednak przy mutacjach losowych (`rand_rewire`) czas wzrasta do około 2.6 sekundy, a koszt również rośnie. Algorytm mrówkowy zapewnia bardzo dobrą jakość przy `pref_pref`, ale działa najwolniej (prawie 7 sekund). Przejście na `pref_triadic` ponad dwukrotnie przyspiesza jego działanie kosztem niewielkiego pogorszenia jakości.

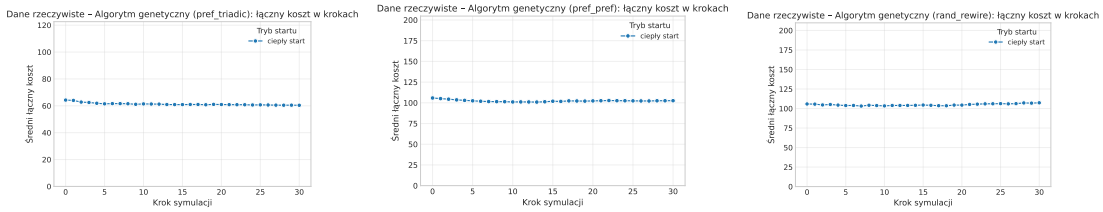
Heurystyki szybkie, takie jak algorytm zachłanny (około 1 ms) i zbiór dominujący (około 5 ms), są bardzo efektywne. Zbiór dominujący zwykle osiąga niższy koszt niż zachłanny, co czyni go dobrym wyborem przy ograniczeniach czasowych. Mutacje klastrowe (`pref_triadic`, `pref_pref`) pomagają wszystkim algorytmom utrzymać niski koszt i krótszy czas. Natomiast `rand_rewire` zwiększa zarówno koszt, jak i czas.

7.4.2 Ewolucja kosztów w czasie

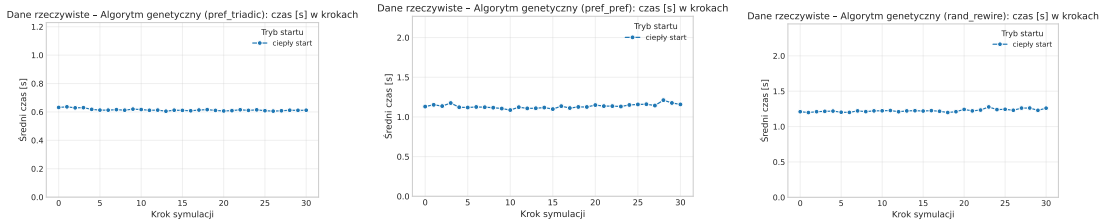
Pełne przebiegi dla algorytmu genetycznego pokazano na rys. 7.3–7.4. Łączenie preferencyjnego przyłączania z triadycznym domykaniem sprzyja utrzymaniu najniższych kosztów, natomiast wariant z losowym przełączaniem krawędzi prowadzi do wolniejszej stabilizacji. Analizując koszt na węzeł (Rys. 7.3), można zauważyć, że dla każdego z typów mutacji przebiega on podobnie, oscylując wokół zbliżonego poziomu.

Z kolei, patrząc na czas wykonania (Rys. 7.4), widać, że dla wariantu `rand_rewire` występują największe wahania, co sugeruje większą niestabilność w procesie optymalizacji. Warianty

`pref_triadic` i `pref_pref` charakteryzują się bardziej stabilnym czasem wykonania.



Rysunek 7.3: Algorytm genetyczny – koszt na węzeł w wariantach realistycznych.



Rysunek 7.4: Algorytm genetyczny – czas wykonania w wariantach realistycznych.

7.4.3 Wnioski

Przeprowadzona analiza dynamiczna wykazała, że dobór algorytmu optymalizacyjnego w środowisku dynamicznym powinien uwzględniać zarówno profil zmian topologii sieci, jak i dostępny budżet czasowy. Badania obejmujące sześć wariantów mutacji (trzy syntetyczne oraz trzy realistyczne) na różnych typach grafów pozwoliły na sformułowanie kluczowych wniosków dotyczących adaptacji algorytmów do ewoluujących struktur sieciowych.

Mutacje realistyczne, takie jak `pref_triadic` oraz `pref_pref`, okazały się łatwiejsze do obsłużenia w porównaniu z wariantami syntetycznymi o wysokiej intensywności. Mechanizmy preferencyjnego przyłączania oraz domykania trójkątów sprzyjają wzmacnianiu klastrowości i tworzeniu hubów, co zmniejsza efektywny rozmiar problemu pokrycia. W rezultacie algorytmy osiągały niższe koszty (0.470–0.475 w porównaniu do 0.485–0.489) oraz krótsze czasy wykonania (0.9–1.7 s w porównaniu do 2.9–3.2 s). Z kolei mutacje typu `rand_rewire` zwiększały entropię topologii poprzez zrywanie lokalnych połączeń, co prowadziło do najgorszych wyników zarówno pod względem kosztu (0.486), jak i czasu (1.9 s).

Algorytmy wykazywały różną odporność na poszczególne typy zmian w zależności od mechanizmu wykorzystania poprzedniego rozwiązania. Solver ILP zapewniał najwyższą jakość (koszt 0.362–0.390), lecz wymagał znacznego czasu na ponowne rozwiązanie zmodyfikowanych ograniczeń. Algorytm genetyczny dobrze radził sobie przy zmianach klastrowych, oferując korzystny kompromis między jakością a czasem (koszt 0.409 przy czasie 0.615 s dla `pref_triadic`), jednak jego efektywność spadała w przypadku mutacji destrukcyjnych. Przeszukiwanie tabu, jako metoda intensywnie lokalna, sprawdzało się przy zmianach o ograniczonym zasięgu, lecz przy mutacjach o większej skali wymagało rozszerzenia promienia poszukiwań, co wydłużało czas działania nawet do 6.5 sekundy.

Szybkie heurystyki, takie jak algorytm zachłanny oraz zbiór dominujący, zachowywały

swoją użyteczność również w środowisku dynamicznym, oferując czasy wykonania poniżej 5 ms przy kosztach porównywalnych z bardziej złożonymi metodami. Algorytm zachłanny osiągał stabilny czas w zakresie 0.5–1.8 ms przy koszcie 0.464–0.480, natomiast zbiór dominujący zapewniał nieco niższy koszt przy czasie około 5 ms.

Wyniki badań wskazują na konieczność adaptacyjnego doboru strategii optymalizacyjnej w zależności od obserwowanego profilu zmian. Przy zmianach o charakterze klastrowym zaleca się stosowanie metaheurystyk z mechanizmami naprawy poprzedniego rozwiązania. W przypadku zmian chaotycznych konieczne może być głębsze odświeżenie rozwiązania lub zaakceptowanie wyższego kosztu w zamian za stabilność czasową. W sytuacjach, w których czas stanowi krytyczne ograniczenie, szybkie heurystyki stanowią bezpieczny wybór, zapewniając przewidywalną wydajność niezależnie od typu mutacji.

Najlepszy kompromis globalny oferują metaheurystyki z mechanizmami konstrukcji opartymi na poprzednich wynikach oraz lokalnych naprawach, które można dynamicznie dostosowywać do intensywności zaobserwowanych zmian między krokami symulacji.

8. ROZSZERZENIA MODELU LICENCJONOWANIA

8.1 Przegląd badanych wariantów

Oprócz bazowych konfiguracji rozważono osiem rozszerzeń licencyjnych, które różnią się wielokrotnością kosztu licencji grupowej względem indywidualnej. Warianty Duolingo Super obejmują plany, w których koszt licencji grupowej wynosi odpowiednio dwukrotność, czterokrotność i pięciokrotność ceny licencji indywidualnej, przy zachowaniu stałej pojemności grupy (6 osób). Podobnie, w przypadku dominowania rzymskiego analizowano konfiguracje, w których koszt licencji grupowej wynosi p-krotność ceny indywidualnej, dla $p \in \{3, 4, 5\}$. Dwa ostatnie warianty odnoszą się do rzeczywistych ofert: Spotify wprowadza plan Duo (pojemność 2) pomiędzy licencją indywidualną a rodzinną, natomiast Netflix oferuje plany dla 1, 2 lub 4 osób.

8.1.1 Warianty rodziny Duolingo

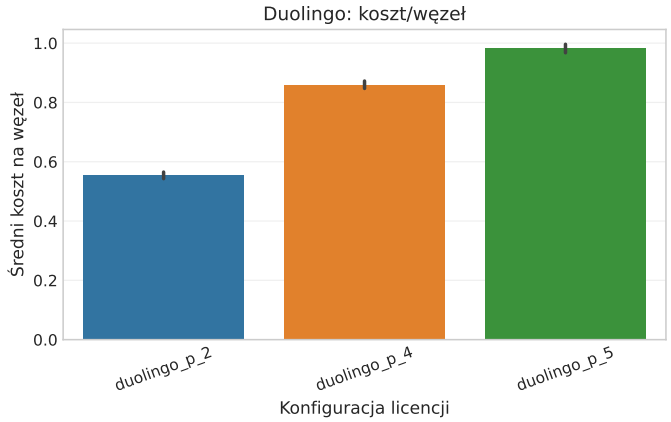
Na rys. 8.1 przedstawiono mediany kosztu na węzeł dla konfiguracji, w których koszt licencji grupowej wynosi odpowiednio dwukrotność (`duolingo_p_2`), czterokrotność (`duolingo_p_4`) oraz pięciokrotność (`duolingo_p_5`) ceny licencji indywidualnej. Widać, że wyższe mnożniki prowadzą do wzrostu kosztu na węzeł, co wynika z preferencji algorytmów do korzystania z licencji indywidualnych przy wyższych kosztach planów grupowych. W przypadku konfiguracji `duolingo_p_4` oznacza to, że sens kupna licencji grupowej pojawia się dopiero dla grup liczących 4 lub więcej osób, ponieważ dopiero wtedy koszt licencji grupowej jest równy lub niższy niż koszt czterech licencji indywidualnych.

Szczegółowe statystyki dla wariantów Duolingo zebrano w tabeli 8.1. Wzrost mnożnika ceny grupowej z 2 do 5 powoduje wzrost średniego kosztu na węzeł o 77% (z 0,554 do 0,982) oraz wydłużenie średniego czasu obliczeń o 47% (z 0,637 s do 0,938 s).

Tabela 8.1: Statystyki dla wariantów Duolingo (benchmark statyczny).

Konfiguracja	Metryka	Średnia	Odch. std.	Min	Max
`duolingo_p_2`	Czas [s]	0.637	1.210	0.000	7.722
	Koszt całkowity	49.152	39.533	8.000	177.000
	Koszt/węzeł	0.554	0.152	0.340	1.000
`duolingo_p_4`	Czas [s]	0.677	1.483	0.000	11.075
	Koszt całkowity	76.815	59.581	14.000	259.000
	Koszt/węzeł	0.860	0.171	0.680	1.520
`duolingo_p_5`	Czas [s]	0.938	2.721	0.000	26.589
	Koszt całkowity	87.855	67.852	17.000	300.000
	Koszt/węzeł	0.982	0.198	0.840	1.820

Analiza rozrzutu wartości w tabeli 8.1 pokazuje stabilność wyników. Odchylenie standardowe dla kosztu na węzeł pozostaje na niskim poziomie (0,152–0,198), co wskazuje na konsekwentność rozwiązań algorytmów w różnych instancjach. Warto zauważyć, że w konfiguracji `duolingo_p_5` odnotowano najwyższy maksymalny czas obliczeń (26,589 s), co może wynikać z większej złożoności problemu przy wyższych kosztach planów grupowych. Zwiększone odchylenie standardowe dla czasów wykonania w tej konfiguracji (2,721 s) sugeruje większą wrażliwość algorytmów na charakterystykę konkretnej instancji sieci.



Rysunek 8.1: Koszt na węzeł w zależności od planu Duolingo (mediany kluczowych algorytmów).

Zmiana struktury licencji wpływa również na udział planów grupowych. W konfiguracji `duolingo_p_2` ponad połowa przydziałów wykorzystuje licencję grupową, podczas gdy w wariancie `duolingo_p_5` udział grup spada do 24%, co odzwierciedla rosnący koszt planu grupowego względem licencji indywidualnych.

8.1.2 Warianty dominowania rzymskiego

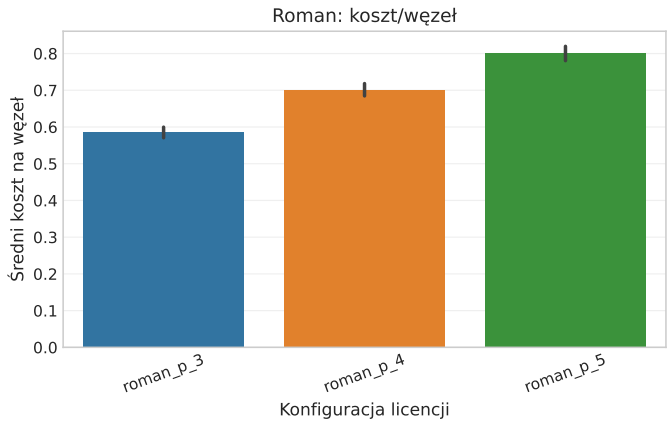
Na rys. 8.2 przedstawiono mediany kosztu na węzeł dla konfiguracji, w których koszt licencji grupowej wynosi odpowiednio trzykrotność (`roman_p_3`), czterokrotność (`roman_p_4`) oraz pięciokrotność (`roman_p_5`) ceny licencji indywidualnej. Podobnie jak w przypadku planów Duolingo, wyższe mnożniki prowadzą do wzrostu kosztu na węzeł, co wynika z preferencji algorytmów do korzystania z licencji indywidualnych przy wyższych kosztach planów grupowych. Wariant `roman_p_5` charakteryzuje się najwyższym kosztem, co odzwierciedla ograniczoną opłacalność licencji grupowych w tej konfiguracji.

Szczegółowe statystyki dla wariantów dominowania rzymskiego zebrano w tabeli 8.2. Wzrost mnożnika ceny grupowej z 3 do 5 powoduje wzrost średniego kosztu na węzeł o 37% (z 0,585 do 0,800) oraz nieznaczne wydłużenie średniego czasu obliczeń (z 0,477 s do 0,493 s).

Tabela 8.2: Statystyki dla wariantów dominowania rzymskiego (benchmark statyczny).

Konfiguracja	Metryka	Średnia	Odch. std.	Min	Max
roman_p_3	Czas [s]	0.477	1.038	0.000	9.522
	Koszt całkowity	50.261	40.552	7.000	218.000
	Koszt/węzeł	0.585	0.199	0.260	1.170
roman_p_4	Czas [s]	0.475	1.125	0.000	9.189
	Koszt całkowity	60.441	48.750	9.000	259.000
	Koszt/węzeł	0.701	0.232	0.340	1.400
roman_p_5	Czas [s]	0.493	1.289	0.000	12.314
	Koszt całkowity	68.995	55.987	11.000	300.000
	Koszt/węzeł	0.800	0.269	0.395	1.660

Zmiana struktury licencji wpływa również na udział planów grupowych (tabela 8.5). W konfiguracji roman_p_3 udział grup wynosi 43%, podczas gdy w wariancie roman_p_5 spada do 24%. Wyższe koszty planów grupowych prowadzą do preferencji algorytmów w kierunku licencji indywidualnych, co jest zgodne z obserwacjami dla innych rozszerzeń.



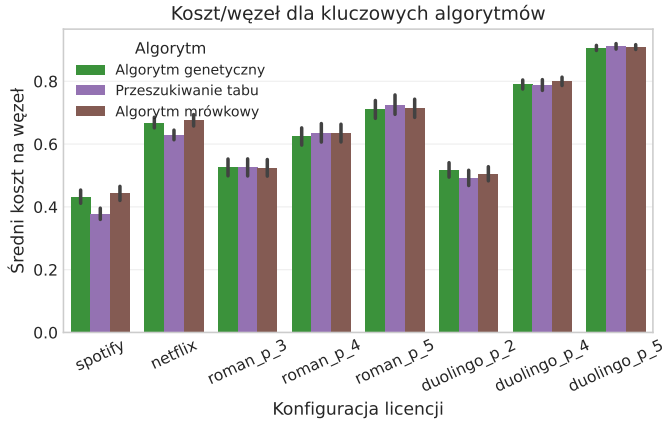
Rysunek 8.2: Koszt na węzeł dla wariantów dominowania rzymskiego.

8.1.3 Spotify i Netflix

Spotify wprowadza trzecią licencję (Duo), co otwiera możliwość parowania użytkowników zamiast budowania dużych grup. Tabela 8.3 potwierdza, że przeszukiwanie tabu i algorytm genetyczny osiągają najniższe koszty ($0{,}335$ i $0{,}400$ na węzeł), wyraźnie dystansując heurystykę zachłanną. W przypadku Netflixa plan Standard (pojemność 2) wypełnia lukę między kontem solo a planem rodzinnych czterech kont, dzięki czemu metaheurystyki utrzymują koszt w granicach $0{,}60$–$0{,}65$ przy czasie poniżej 1 s.

Tabela 8.3: Medians dla konfiguracji Spotify i Netflix.

Konfiguracja	Algorytm	Med. koszt/węzeł	Med. czas [s]
spotify	Algorytm zachłanny	0.446	0.000
spotify	Algorytm genetyczny	0.400	0.229
spotify	Algorytm mrówkowy	0.391	1.040
spotify	Przeszukiwanie tabu	0.335	0.958
netflix	Algorytm zachłanny	0.682	0.000
netflix	Algorytm genetyczny	0.650	0.237
netflix	Algorytm mrówkowy	0.656	1.289
netflix	Przeszukiwanie tabu	0.604	0.999



Rysunek 8.3: Koszt na węzeł dla wszystkich rozszerzeń (mediany).

8.1.4 Porównanie wszystkich rozszerzeń

Podstawowe statystyki dla wszystkich konfiguracji zestawiono w tabeli 8.4. W każdym przypadku analizowano ten sam zestaw algorytmów, pomijając obserwacje z timeout.

Tabela 8.4: Statystyki agregowane dla rozszerzeń (benchmark statyczny).

Konfiguracja	Śr. koszt/węzeł	Śr. czas [s]
duolingo_p_2	0.554	0.637
duolingo_p_4	0.860	0.677
duolingo_p_5	0.982	0.938
spotify	0.479	0.547
netflix	0.729	0.623
roman_p_3	0.585	0.477
roman_p_4	0.701	0.475
roman_p_5	0.800	0.493

Analiza agregowanych statystyk w tabeli 8.4 ujawnia wyraźne różnice między rodzinami konfiguracji. W przypadku wariantów Duolingo obserwuje się silną korelację między mnożnikiem

ceny grupowej a kosztem na węzeł – wzrost parametru z 2 do 5 prowadzi do zwiększenia kosztu o 77% (z 0,554 do 0,982). Jednocześnie czas obliczeń wydłuża się o 47%, co wskazuje na rosnącą złożoność problemu przy wyższych kosztach planów grupowych.

Warianty dominowania rzymskiego charakteryzują się większą stabilnością obliczeniową, z czasami wykonania w zakresie 0,477–0,493 s niezależnie od parametru p. Wzrost kosztu na węzeł jest bardziej umiarkowany (37% przy wzroście p z 3 do 5), co sugeruje odmienną strukturę przestrzeni rozwiązań w porównaniu z planami Duolingo.

Najkorzystniejszy stosunek kosztu do wydajności obliczeniowej wykazuje konfiguracja Spotify (0,479 kosztu na węzeł przy 0,547 s), co potwierdza skuteczność wprowadzenia planu pośredniego (Duo) między licencjami indywidualnymi a rodzinnymi. Netflix zajmuje pozycję pośrednią z kosztem 0,729 na węzeł, pozostając jednak konkurencyjny pod względem czasu obliczeń (0,623 s).

8.1.5 Analiza wpływu liczby użytkowników na koszty

Struktura wykorzystania licencji zmienia się znacząco (tabela 8.5). W Spotify licencje grupowe odpowiadają za 64% przydziałów (plan Duo + rodzina), podczas gdy w wariancie `duolingo_p_5` udział grup spada do 24%. W Netflixie większość przydziałów to plany Standard/Premium (udział 68%).

Tabela 8.5: Udział licencji grupowych i indywidualnych (benchmark statyczny).

Konfiguracja	Udział grup	Udział indywidualnych
duolingo_p_2	0.56	0.44
duolingo_p_4	0.34	0.66
duolingo_p_5	0.24	0.76
spotify	0.64	0.36
netflix	0.68	0.32
roman_p_3	0.43	0.57
roman_p_4	0.34	0.66
roman_p_5	0.24	0.76

8.2 Rozszerzenia w środowisku dynamicznym

W środowisku dynamicznym przeanalizowano te same osiem konfiguracji rozszerzeń, stosując identyczną metodologię jak w benchmarku statycznym. Każda konfiguracja została przetestowana na 272 instancjach dynamicznych, obejmujących różne rozmiary sieci i scenariusze zmian.

8.2.1 Statystyki agregowane

Tabela 8.6 przedstawia statystyki zagregowane według rodzin konfiguracji. Warianty Duolingo charakteryzują się najwyższymi kosztami średnimi (73,83 na instancję) oraz najdłuższymi czasami wykonania (0,97 s), co wynika z konieczności rozważania licencji o pojemności 6

osób. Rodzina Roman wykazuje umiarkowane koszty (59,96) przy porównywalnym czasie obliczeń (0,85 s). Konfiguracje rzeczywistych serwisów – Spotify i Netflix – osiągają najkorzystniejsze wyniki, z najniższymi kosztami odpowiednio 43,32 i 66,58.

Tabela 8.6: Statystyki zagregowane według rodzin konfiguracji (benchmark dynamiczny).

Rodzina	Śr. czas [s]	Śr. koszt całk.	Śr. koszt/węzeł	Liczba obs.
Duolingo	0.969	73.83	0.792	3265
Roman	0.853	59.96	0.661	3408
Netflix	0.891	66.58	0.714	1074
Spotify	0.881	43.32	0.467	1088

8.2.2 Porównanie szczegółowe konfiguracji

Tabela 8.7 zawiera szczegółowe statystyki dla wszystkich wariantów rozszerzeń. W rodzinie Duolingo obserwuje się wyraźny wzrost kosztu na węzeł wraz ze wzrostem mnożnika ceny grupowej – od 0,539 w konfiguracji `duolingo_p_2` do 0,980 w `duolingo_p_5`. Podobną tendencję wykazują warianty dominowania rzymskiego, gdzie koszt rośnie z 0,554 (`roman_p_3`) do 0,763 (`roman_p_5`).

Tabela 8.7: Szczegółowe statystyki dla rozszerzeń (benchmark dynamiczny).

Konfiguracja	Śr. czas [s]	Śr. koszt całk.	Śr. koszt/węzeł	Liczba obs.
duolingo_p_2	0.939	50.22	0.539	1088
duolingo_p_4	0.987	80.17	0.855	1073
duolingo_p_5	0.980	90.94	0.980	1104
spotify	0.881	43.32	0.467	1088
netflix	0.891	66.58	0.714	1074
roman_p_3	0.798	50.30	0.554	1136
roman_p_4	0.930	60.41	0.666	1136
roman_p_5	0.829	69.18	0.763	1136

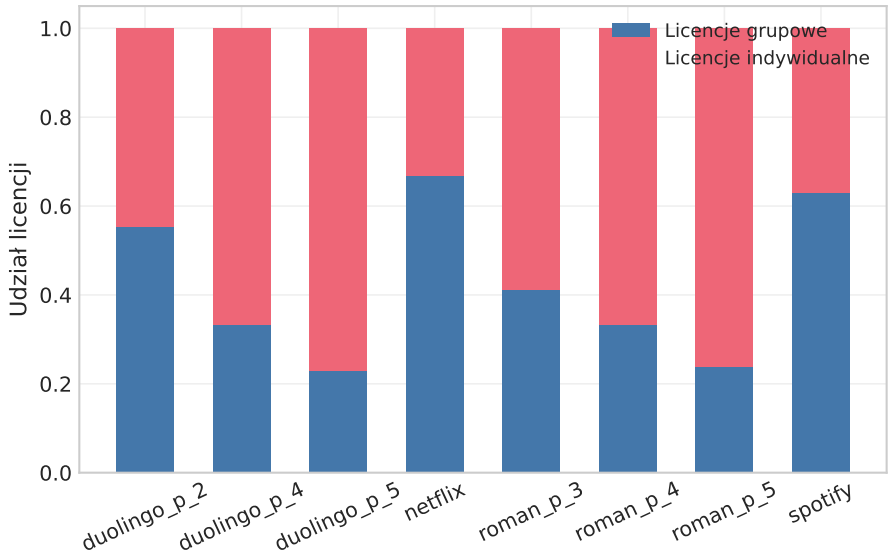
Konfiguracja Spotify potwierdza swoją przewagę osiągając najniższy koszt na węzeł (0,467), co stanowi wynik o 13% lepszy niż w najbliższej konfiguracji `roman_p_3`. Plan Duo umożliwia efektywne parowanie użytkowników, co przekłada się na oszczędności w całym spektrze rozmiarów sieci. Netflix zajmuje pozycję pośrednią z kosztem 0,714 na węzeł, pozostając konkurencyjny wobec wariantów o wysokich mnożnikach.

8.2.3 Struktura wykorzystania licencji

Analiza składu licencji (tabela 8.8) ujawnia znaczące różnice w strategiach przydzielania. Konfiguracje o niskich mnożnikach (`duolingo_p_2`, `spotify`) preferują licencje grupowe, osiągając udział odpowiednio 55% i 63%. W przeciwieństwie do tego, wysokie koszty planów grupowych w `duolingo_p_5` i `roman_p_5` prowadzą do dominacji licencji indywidualnych (76% i 76% udziału).

Tabela 8.8: Struktura wykorzystania licencji (benchmark dynamiczny).

Konfiguracja	Udział grup [%]	Udział indywidualnych [%]
duolingo_p_2	55.4	44.6
duolingo_p_4	33.3	66.7
duolingo_p_5	23.0	77.0
spotify	63.0	37.0
netflix	66.8	33.2
roman_p_3	41.1	58.9
roman_p_4	33.4	66.6
roman_p_5	23.8	76.2



Rysunek 8.4: Udział licencji grupowych i indywidualnych w rozszerzeniach dynamicznych.

Netflix wykazuje najwyższy udział planów grupowych (67%), co wynika z atrakcyjności planów Standard i Premium dla grup 2-4 osobowych. Ta konfiguracja skutecznie wypełnia lukę między licencjami indywidualnymi a planami o dużej pojemności, umożliwiając algorytmom elastyczne dopasowanie do struktury sieci.

8.2.4 Porównanie z benchmarkiem statycznym

Zestawienie wyników dynamicznych ze statycznymi (tabela 8.9) pokazuje generalną zgodność trendów. Koszty w środowisku dynamicznym pozostają na podobnym poziomie – różnice nie przekraczają 3% dla większości konfiguracji. Czasy wykonania wydłużają się o 30-60%, co odzwierciedla dodatkową złożoność związaną z dynamicznym rebalansowaniem.

Tabela 8.9: Porównanie wyników statycznych i dynamicznych.

Konfiguracja	Koszt stat.	Koszt dyn.	Czas stat.	Czas dyn.
duolingo_p_2	0.554	0.539	0.637	0.939
duolingo_p_4	0.860	0.855	0.677	0.987
duolingo_p_5	0.982	0.980	0.938	0.980
spotify	0.479	0.467	0.547	0.881
netflix	0.729	0.714	0.623	0.891
roman_p_3	0.585	0.554	0.477	0.798
roman_p_4	0.701	0.666	0.475	0.930
roman_p_5	0.800	0.763	0.493	0.829

Najstabilniejsze wyniki wykazuje konfiguracja `duolingo_p_5`, gdzie koszty różnią się jedynie o 0,002 między benchmarkami. Największą poprawę w środowisku dynamicznym odnotowano dla `roman_p_4` (redukcja kosztu o 5%) i `roman_p_5` (redukcja o 4,6%). Może to wynikać z lepszego wykorzystania możliwości rebalansowania w mniejszych grupach charakterystycznych dla tych konfiguracji.

8.3 Wnioski

Przeprowadzone badania nad rozszerzeniami modelu licencjonowania pozwoliły na kompleksową analizę wpływu struktury cenowej na optymalizację kosztów w sieciach społecznych. Kluczowe wnioski z analizy ośmiu wariantów licencyjnych można podzielić na trzy główne obszary.

Wpływ mnożnika ceny grupowej na efektywność algorytmów. Badania wykazały silną zależność między stosunkiem ceny licencji grupowej do indywidualnej a skutecznością optymalizacji. W wariantach Duolingo wzrost mnożnika z 2 do 5 powoduje zwiększenie średniego kosztu na węzeł o 77% (z 0,554 do 0,982) oraz wydłużenie czasu obliczeń o 47%. Podobną, choć łagodniejszą tendencję obserwuje się w wariantach dominowania rzymskiego, gdzie wzrost parametru p z 3 do 5 prowadzi do 37% wzrostu kosztu przy stabilnym czasie wykonania. Te wyniki wskazują, że metaheurystyki osiągają największe korzyści przy niskich mnożnikach, gdzie licencje grupowe pozostają atrakcyjne ekonomicznie.

Znaczenie planów pośrednich w strukturze licencji. Analiza konfiguracji rzeczywistych serwisów ujawniła istotną rolę licencji o pojemności pośredniej. Spotify z planem Duo (pojemność 2) osiąga najkorzystniejszy stosunek kosztu do wydajności (0,467 kosztu na węzeł przy 0,547s czasu), podczas gdy Netflix z planem Standard oferuje skuteczne wypełnienie luki między licencjami indywidualnymi a rodzinnymi. W obu przypadkach udział licencji grupowych przekracza 63%, co potwierdza efektywność elastycznego spektrum opcji licencyjnych.

Stabilność wyników w środowisku dynamicznym. Porównanie benchmarków statycznego i dynamicznego wykazało wysoką zgodność rezultatów – różnice w kosztach na węzeł nie przekraczają 3% dla większości konfiguracji. Jednocześnie zaobserwowano wzrost czasów wykonania

o 30-60%, co odzwierciedla dodatkową złożoność związaną z dynamicznym rebalansowaniem. Najstabilniejsze wyniki uzyskano w konfiguracji `duolingo_p_5`, gdzie różnica między środowiskami wynosi jedynie 0,002. Te obserwacje potwierdzają, że proponowane rozszerzenia modelu zachowują skuteczność w realistycznych scenariuszach z czasowymi zmianami struktury sieci.

Otrzymane wyniki mają istotne implikacje praktyczne dla projektowania systemów licencjonowania. Po pierwsze, wprowadzenie planów o pojemności pośredniej może znacząco poprawić efektywność kosztową bez zwiększenia złożoności obliczeniowej. Po drugie, przy wysokich mnożnikach ceny grupowej (powyżej 4-5) korzyści z metaheurystyk maleją, co sugeruje ekonomiczne ograniczenia optymalizacji algorytmicznej. Po trzecie, stabilność wyników w środowisku dynamicznym wskazuje na praktyczną stosowalność proponowanych rozwiązań w rzeczywistych systemach z fluktuacją użytkowników.

9. PODSUMOWANIE

9.0.1 Symulacje dynamiczne

Rozdział 7 opisuje symulator zmian w sieci. Uwzględnia mutacje węzłów i krawędzi oraz trzy tryby: preferencyjny, triadyczny i losowy. Zastosowanie ciepłego startu, czyli użycie poprzedniego rozwiązania, obniżało koszt względem liczenia od zera o 6–14% na grafach syntetycznych i o 7–12% na danych realistycznych. Czas ponownego zrównoważenia wynosił zwykle 1–3 s. Algorytm zachłanny działał w milisekundach i służył jako punkt odniesienia. Intensywność mutacji silniej wpływała na czas niż na końcowy koszt. Najwyższą dokładność uzyskiwał algorytm mrówkowy, a przeszukiwanie tabu i algorytm genetyczny dawały lepszy kompromis między kosztem i czasem. W ujęciu dynamicznym najtrudniejszy był wariant `rand_rewire`.

Praca pokazuje pełny cykl badawczy dotyczący optymalizacji kosztów licencji grupowych w sieciach społecznościowych. Najpierw sformalizowano model. Następnie porównano algorytmy deterministyczne i metaheurystyczne. Dalej przeprowadzono symulacje dynamiczne. Na końcu rozszerzono analizę o dodatkowe plany licencyjne. Uzyskano spójny obraz działania metod w wielu scenariuszach. Poniżej zebrano główne wyniki i wskazano możliwe kierunki dalszych badań.

9.1 Wyniki

9.1.1 Model i metody

Rozdziały 1–4 definiują problem jako uogólnienie dominacji rzymskiej z ograniczeniami pojemności oraz różnymi typami licencji. Wykazano wysoką złożoność obliczeniową. Rozdział 5 prezentuje pełen zestaw metod: dokładny solver ILP, heurystyki konstrukcyjne (m.in. algorytm zachłanny, podejście przez zbiór dominujący), metaheurystyki (algorytm genetyczny, algorytm mrówkowy, przeszukiwanie tabu, wyżarzanie symulowane) oraz algorytm losowy. Wszystkie implementacje mają wspólny interfejs, co ułatwiło porównania.

9.1.2 Eksperymenty statyczne

Rozdział 6 potwierdza hierarchię jakości: $ILP >$ algorytm mrówkowy $>$ przeszukiwanie tabu/algorytm genetyczny $>$ wyżarzanie symulowane $>$ heurystyki konstrukcyjne $>$ algorytm losowy. Różnice są istotne statystycznie (test Friedmana oraz porównania Nemenyi'ego). Grafy bezskalowe okazały się łatwiejsze do pokrycia z powodu hubów. Czas działania metaheurystyk rósł szybko, a heurystyki konstrukcyjne utrzymywały czasy rzędu milisekund. W praktyce wyróżniono trzy zakresy: dla małych grafów warto stosować ILP jako punkt odniesienia; dla średnich grafów najlepsze są metaheurystyki; dla dużych grafów albo gdy liczy się szybkość, użyteczną aproksymację daje algorytm zachłanny.

9.1.3 Symulacje dynamiczne

Rozdział 7 przedstawia wpływ zmian w sieci na koszt i czas. Ciepły start konsekwentnie obniżał koszt o 6–14% (syntetyczne) i 7–12% (realistyczne), przy czasie rebalansowania 1–3 s. Intensywniejsze mutacje zwiększały głównie czas, a mniej wpływały na koszt. Najdokładniejszy pozostawał algorytm mrówkowy, natomiast przeszukiwanie tabu i algorytm genetyczny dawały lepszy kompromis czasowy. W danych realistycznych najniższe koszty uzyskano w wariancie Spotify (mediana $\approx 0{,}41$ na węzeł).

9.1.4 Rozszerzenia licencyjne

Rozdział 8 analizuje osiem wariantów taryf. W planach Duolingo i Roman o opłacalności decydowały rozmiar grupy i koszt planu. Przy tańszej grupie (`duolingo_p_2`) metaheurystyki redukowały koszt o 10–20% względem heurystyk. Przy droższych planach (`duolingo_p_5`, `roman_p_5`) przewaga spadała do kilku procent. Dodanie planu Duo (Spotify) oraz planów Standard/Premium (Netflix) umożliwiło nowe parowania użytkowników i obniżyło koszt na węzeł (mediana 0,40 w Spotify wobec 0,50 w `duolingo_p_2`). W dynamice konfiguracje z planem pośrednim utrzymywały najkorzystniejszy koszt i stabilny czas, a metaheurystyki dawały 5–12% oszczędności względem algorytmu zachłannego.

9.2 Rekomendacje praktyczne

Dobór algorytmu. Dla instancji do około 200 węzłów solver ILP jest najlepszym punktem odniesienia. Dla większych sieci zalecane są algorytm mrówkowy albo przeszukiwanie tabu; w środowisku dynamicznym krótsze czasy rebalansowania daje przeszukiwanie tabu.

Strategia inicjalizacji. Ciepły start metaheurystyk, czyli start z poprzedniego rozwiązania, obniża koszt o 6–14% przy niewielkim koszcie czasowym. W systemach aktualizowanych częściowo powinien to być standard.

Polityka licencyjna. Tańsze plany rodzinne, w szczególności warianty pośrednie (np. Duo), realnie zmniejszają koszt końcowy. Zbyt wysokie p w planach grupowych sprzyja licencjom indywidualnym i ogranicza zyski z optymalizacji.

9.3 Kierunki dalszych badań

Gwarancje aproksymacji. Warto poszukać teoretycznych ograniczeń jakości rozwiązań dla wybranych klas grafów lub modeli losowych. **Modele stochastyczne.** Ujęcie niepewności w dostępie użytkowników i ewolucji sieci może zwiększyć odporność rozwiązań (np. podejścia dwuetapowe lub bayesowskie). **Wielokryterialność i koszty operacyjne.** Rozszerzenie funkcji celu o sprawiedliwość, ryzyko czy koszt migracji lepiej odzwierciedli praktykę. **Optymalizacja hiperparametrów.** Automatyczne strojenie parametrów metaheurystyk (np. techniki z rodziny AutoML) może poprawić stosunek kosztu do czasu bez ręcznego dostrajania. **Integracja z praktyką.** Biblioteka daje podstawy do wdrożeń w systemach rekomendacji planów rodzinnych; naturalnym krokiem jest eksperyment online na rzeczywistych danych.

9.4 Zakończenie

Przedstawiony model, zestaw algorytmów oraz eksperymenty statyczne i dynamiczne pokazują, że mimo wysokiej złożoności można uzyskać rozwiązania dobrej jakości w akceptowalnym czasie. Kluczowy jest dobór metody do wielkości i dynamiki sieci oraz rozsądna polityka licencyjna. Wyniki stanowią podstawę do dalszych prac oraz do zastosowań w usługach subskrypcyjnych, gdzie modele rodzinne stają się standardem.

WYKAZ LITERATURY

1. *Subscription Economy Index* [Industry report]. 2024. Dostępne także z: https://www.zuora.com/resource/subscription-economy-index/.
2. *Subscription Price Trends 2024* [Industry report]. 2024. Dostępne także z: https://recurly.com/press/recurly-releases-its-2024-state-of-subscriptions-report/.
3. *Duolingo Family Plan* [Product page]. 2024. Dostępne także z: https://support.duolingo.com/hc/pl/articles/6159959913243-Plan-rodzinny-Super-Duolingo.
4. BRANDES, Ulrik; ERLEBACH, Thomas (red.). *Network Analysis: Methodological Foundations*. T. 3418. Springer, 2005. Lecture Notes in Computer Science. Dostępne z DOI: 10.1007/978-3-540-31955-9.
5. NETTLETON, David F. Data mining of social networks represented as graphs. *Computer Science Review*. 2013, t. 7, nr. 1, s. 1–34. Dostępne z DOI: 10.1016/j.cosrev.2012.12.001.
6. SPOTIFY POLSKA. *Spotify Premium - Plany i ceny (Polska)*. 2025. Dostępne także z: https://www.spotify.com/pl/premium/. Stan na 09.2025: Individual - 23,99 PLN, Duo - 30,99 PLN, Family - 37,99 PLN miesięcznie.
7. SPOTIFY POLSKA. *Spotify Premium – Plany i ceny (Polska)*. 2025. Dostępne także z: https://www.spotify.com/pl/premium/. Stan na 09.2025: Basic – 33,00 PLN, Standard – 49,00 PLN, Premium – 67,00 PLN miesięcznie.
8. DUOLINGO. *Super Duolingo - ceny w aplikacji mobilnej (Polska)* [Aplikacja mobilna Duolingo, wersja na Android]. 2025. Stan na 09.2025: Individual - 13,99 PLN, Family - 29,17 PLN miesięcznie.
9. HAYNES, Teresa W.; HEDETNIEMI, Stephen T.; SLATER, Peter J. *Fundamentals of Domination in Graphs*. Marcel Dekker, 1998.
10. *Dominating set* [Wikipedia]. 2024. Dostępne także z: https://en.wikipedia.org/wiki/Dominating_set.
11. POUREIDI, Abolfazl; FATHALI, Jafar. Algorithmic results in Roman dominating functions on graphs. *Information Processing Letters*. 2023, t. 182, s. 106363. Dostępne z DOI: 10.1016/j.ipl.2023.106363.
12. PANDA, B. S.; RANA, Soumyashree; MISHRA, Sounaka. On the complexity of co-secure dominating set problem. *Information Processing Letters*. 2024, t. 185, s. 106463. Dostępne z DOI: 10.1016/j.ipl.2023.106463.
13. ALIMONTI, Paola; KANN, Viggo. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*. 2000, t. 237, nr. 1–2, s. 123–134. Dostępne z DOI: 10.1016/S0304-3975(99)00426-0.

14. BERMAN, Piotr; FUJITO, Toshihiro. On the Approximation Properties of the Independent Set Problem in Degree 3 Graphs. W: *Algorithms and Data Structures*. Springer, 1995, t. 955, s. 449–460. Lecture Notes in Computer Science. ISBN 978-3-540-60165-6.
15. FAVARON, Odile; KARAMI, Hosein; KHOEILAR, Reza; SHEIKHOLESLAMI, Seyed Mahmoud. On the Roman domination number of a graph. *Discrete Mathematics*. 2009, t. 309, nr. 10, s. 3447–3451. Dostępne z DOI: 10.1016/j.disc.2008.09.043.
16. COCKAYNE, Ernie J.; DREYER Paul A., Jr.; HEDETNIEMI, Sandra M.; HEDETNIEMI, Stephen T. Roman domination in graphs. *Discrete Mathematics*. 2004, t. 278, nr. 1–3, s. 11–22. Dostępne z DOI: 10.1016/j.disc.2003.06.004.
17. CHAUDHARY, Juhi; PRADHAN, Dinabandhu. Roman 3-domination in graphs: Complexity and algorithms. *Discrete Applied Mathematics*. 2024, t. 354, s. 301–325. Dostępne z DOI: 10.1016/j.dam.2022.09.017.
18. GHAFFARI-HADIGHEH, Alireza. Roman domination problem with uncertain positioning and deployment costs. *Soft Computing*. 2019, t. 24, nr. 4, s. 2637–2645. Dostępne z DOI: 10.1007/s00500-019-03811-z.
19. CHAMBERS, Erin W.; KINNERSLEY, Bill; PRINCE, Noah; WEST, Douglas B. Extremal problems for Roman domination. *SIAM Journal on Discrete Mathematics*. 2009, t. 23, nr. 3, s. 1575–1596. Dostępne z DOI: 10.1137/070699688.
20. KUHN, Fabian. *Network Algorithms (Graduate Course) – Lecture Notes* [Course notes]. 2012. Dostępne także z: https://algo.inf.uni-freiburg.de/teaching/ss12/network-algorithms/. Graduate course at the University of Freiburg, Summer Term 2012.
21. PARRA INZA, Ernesto; VAKHANIA, Nodari; SIGARRETA ALMIRA, Jose Maria; HERNANDEZ-AGUILAR, Jose Alberto. Approximating a Minimum Dominating Set by Purification. *Algorithms*. 2024, t. 17, nr. 6, s. 258. Dostępne z DOI: 10.3390/a17060258.
22. ERDŐS, Paul; RÉNYI, Alfréd. On the evolution of random graphs. *Publicationes Mathematicae*. 1960, t. 5, s. 17–61.
23. BARABÁSI, Albert-László; ALBERT, Réka. Emergence of scaling in random networks. *Science*. 1999, t. 286, nr. 5439, s. 509–512. Dostępne z DOI: 10.1126/science.286.5439.509.
24. WATTS, Duncan J.; STROGATZ, Steven H. Collective dynamics of small-world networks. *Nature*. 1998, t. 393, s. 440–442. Dostępne z DOI: 10.1038/30918.
25. PROJECT, Stanford Network Analysis. *SNAP Datasets* [https://snap.stanford.edu/data/]. 2024. Dostęp: 2025-09.
26. MCAULEY, Julian; LESKOVEC, Jure. Learning to Discover Social Circles in Ego Networks. W: *Advances in Neural Information Processing Systems 25*. 2012, s. 548–556. Dostępne także z: https://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks. Proceedings of NeurIPS 2012.
27. UGANDER, Johan; KARRER, Brian; BACKSTROM, Lars; MARLOW, Cameron. *The Anatomy of the Facebook Social Graph* [arXiv preprint arXiv:1111.4503]. 2011. Dostępne także z: https://arxiv.org/abs/1111.4503.

28. HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
29. GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
30. GLOVER, F. Tabu Search–Part I. *ORSA Journal on Computing*. 1989, t. 1, nr. 3, s. 190–206. Dostępne z DOI: 10.1287/ijoc.1.3.190.
31. DORIGO, M.; GAMBARDELLA, L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*. 1997, t. 1, nr. 1, s. 53–66. Dostępne z DOI: 10.1109/4235.585892.
32. KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. *Science*. 1983, t. 220, nr. 4598, s. 671–680. Dostępne z DOI: 10.1126/science.220.4598.671.